Thesis/
Reports
Bevins,
C. D.

A PROTOTYPE MODEL FOR THE ASSESMENT OF CUMULATIVE EFFECTS ON GRIZZLY BEARS

June 12, 1992

CO-OP AGREEMENT
Systems For
Environmental
Management

FINAL REPORT FOR RESEARCH JOINT VENTURE AGMT
#INT-90490-RJVA with
SYSTEMS FOR ENVIRONMENTAL MANAGEMENT
FS Contact L. Jack Lyon
Co-op Contact: Collin D. Bevins

A Prototype Model for the Assessment of Cumulative Effects on Grizzly Bears

FINAL REPORT
INT-90490-RJVA
June 12, 1992

Submitted by
Collin D. Bevins
Systems for Environmental Management

INTERMOUNTAIN RESEARCH STATION
JUN 22 1992
RECEIVED

This is the final report to Research Joint Venture Agreement INT-90490-RJVA between the Intermountain Research Station, Ogden, UT, and Systems for Environmental Management, Missoula, MT. In accordance with the objectives set forth in the agreement, the following tasks were completed:

1 A prototype implementation of the unified cumulative effects model for grizzly bears was developed in full accordance with USDA Forest service guidelines. The model integrates spatial vegetation-habitat map data and grizzly bear use preference data with expected impacts from various human activities to assess the resultant reduction in habitat effectiveness.

2 The model consists of a tool kit of 22 command-driven data analysis and evaluation programs, collectively known as "The ICE Tool Kit". They are written in ANSI-standard C, meet federal POSIX requirements, and use the ORACLE 5.1 data base management system. The program run on MS-DOS personal computers, produce standard ASCII text output, and will run under X windowing systems.

3 The ICE Tool Kit uses map information that can be generated from MOSS, Arc/Info, GRASS, or ERDAS geographic information systems.

4 Four meetings were held with personnel from the Flathead National Forest and Montana Dept of Fish, Wildlife and Parks, as well as the Northern Continental Divide West technical committee. However, since their data collection and analysis was lagging behind model development, the actual model training, testing, and evaluation was conducted on the Gallatin National Forest for the Madison Bear Management Subunit 2 and the Mosquito-Denny IRA, and on the Targhee National Forest for the Bechler-Teton Bear Management Unit.

5 Training of Gallatin and Targhee NF personnel was conducted in Missoula, MT, Bozeman, MT, and St. Anthony, ID. Demonstration sessions were also conducted for inter-agency personnel in Missoula, MT.

6 Telephone support was provided on an as-needed basis to the testing sites throughout the agreement period.

7 The ICE Tool Kit User Guide was written and distributed to the user community, and is appended to this report.

The ICE Tool Kit
for
Cumulative Effects Modeling

Beta Test Version
June 1991

PRODUCT WARNING !

This software and user manual is a 100% matter product. In the unlikely event that this merchandise should contact antimatter in any form, a catastrophic explosion will result.

0. PREFACE TO ALPHA VERSION

Gentle Reader:

It is my intent that *The ICE Tool Kit User's Guide* eventually becomes the repository for darn near everything anyone needs to know about CEM. CEM analyses of old relied far too heavily on folklore and oral tradition. I hope the completed manuscript will be distributed in 3-ring binders to which you will add your own notes and parameter tables.

I would like the first chapter in particular to be a candid assessment of the task at hand and discussion of what CEM can and cannot do. A reader new to CEA should be able to read the first 10 to 20 pages, make an accurate assessment of the task at hand, and present that assessment to supervisors.

I am releasing this very preliminary draft for two reasons. The first is to make data preparation guidelines available to you who are already in this phase. The second is to solicit your insights and experiences in building Chapter 1. What would you have liked to have known before you started CEM?

I invite you to make comments, corrections, deletions, and additions. Is the presentation too light, flippant, sarcastic, cynical, or boring? If you'd like to contribute to the next draft, please return your comments to Rodd Richardson by June 1, 1991.

With your input, I may be able to avoid reviewer Moses Hadas' (1900-1966) damnation:

"Thank you for sending me a copy of your book. I'll waste no time reading it."

1. GETTING STARTED

The Incremental Cumulative Effects (ICE) Tool Kit is a collection of computer programs to assist you in performing a cumulative effects analysis (CEA). It is a software implementation of the cumulative effects model described in "*CEM - A Model for Assessing Effects on Grizzly Bears*" (USDA Forest Service, *et.al.*, 1990). This document, also known as the "Green Book", describes cumulative effects as:

> "... the combined effects upon a species or its habitat caused by the activity or program at hand, as well as other reasonably foreseeable events which are likely to have similar effects upon that species or its habitat. Cumulative effects can result from individually minor but collectively significant events taking place over a period of time."

Mattson and Knight (1991) recommend a four-phase cumulative effects analysis process for threatened and endangered species, including (1) a population viability analysis, (2)a habitat capability analysis, (3) the establishment of thresholds, and (4) an analysis of distribution of effects. Past CEM attempts have focused on the second phase.

The ICE Tool Kit has two general purposes. First, it maintains a repository of cumulative effects parameters, maps, and schedules for one or more projects. Second, it computes habitat value (HV), habitat effectiveness (HE), and a mortality risk index (MRI).

While the concepts behind CEM appear straightforward, past implementations of the model have proven difficult and time-consuming to use. The variety of data sources and parameter types, combined with the sheer volume of information to be processed, make CEM a daunting challenge regardless of the hardware and software brought to bear.

The objective of this "*User's Guide*" is to present the CEM and the ICE Tool Kit in a progressive, linear manner. We begin with the basic concepts, work through the various CEM steps using a simple example, and conclude with technical references to programs and data structures. The appendices contain additional information on installation and ecosystem-specific topics.

Before proceeding further, it is important you understand that CEM remains a time-consuming task. You will need to read and comprehend most, and possibly all, of this manual. CEM will still take three times longer than you planned, but hopefully with less confusion and fewer dead ends than experienced by those pioneers who preceded you.

1.1 The Cumulative Effects Model Revisited

If you don't surf, don't start.
- *Tee shirt seen at Clearwater Beach, Fla.*

Despite the sagacity of this advice, let's dive right in. The purpose of the cumulative effects model is to quantify three measures of grizzly bear habitat:

1 the underlying **habitat value** (HV) of an area to the grizzly population,

2 the **habitat effectiveness** (HE) of the underlying habitat resulting from roads, trails, forest management, residential-commercial-recreational development, and other human activities present in the area, and

3 the impact of these activities on grizzly bear mortality through the **mortality risk index** (MRI).

Habitat Value

Let's consider an acre of grizzly bear habitat containing a uniform vegetation type. The vegetation type has a coefficient between zero and one reflecting its importance to bears as interpreted by wildlife biologists. Zero means the biologist thinks the vegetation provides no value to the bear. One means the biologist thinks the bear really likes the stuff. The **vegetative coefficient** may vary season to season or even day to day.

If today is July 4, and the vegetative coefficient (VC) for today is 0.5 (pretty darn good), then the habitat value (so far) is:

$$HV_{July\ 4} = VC_{July\ 4}$$

$$HV_{July\ 4} = 0.5$$

The vegetation also has a cover attribute. Some vegetative types offer hiding or **security cover** to bears, and other vegetative types do not. Let's assume our little acre does provide adequate security cover. If an area without adequate cover is within a certain distance (the **edge buffer zone**) of our area, then our acre is in an **edge buffer**. Edge buffers are highly valued by wildlife biologists, so the vegetative value of our acre is multiplied by an **edge coefficient**. Edge coefficients, like vegetative coefficients, may vary season to season or even day to day.

If the edge coefficient (EC) for the forested side of an edge buffer on July 4 is 2.0, the habitat value of our little acre becomes:

$$HV_{July\ 4} = VC_{July\ 4} \times EC_{July\ 4}$$

$$HV_{July\ 4} = 0.5 \times 2.0 = 1.0$$

Occasionally there are additional factors supplementing the habitat value of a location, increasing its importance to the bear. The habitual presence of ungulates, for example, may draw bears to our acre irrespective of the vegetation. Supplemental factors could include denning sites, travel corridors, bone yards, and so forth. As usual, **supplement coefficients** may vary throughout the year.

If our sample acre is within elk summer range, and the supplement coefficient (SC) for July 4 is 2.0, our final habitat value is:

$$HV_{July\ 4} = VC_{July\ 4} \times EC_{July\ 4} \times SC_{July\ 4}$$

$$HV_{July\ 4} = 0.5 \times 2.0 \times 2.0 = 2.0$$

Habitat Effectiveness

Normally, a forested acre with elk and a nearby meadow (e.g., with a habitat value of 2.0) on July 4 would be pretty attractive to a bear. However, highway US 89 runs along the east edge of the property. Furthermore, our sample acre is smack in the middle of a 10 vehicle campground. This will likely reduce the effectiveness of the underlying habitat.

While US 89 doesn't run right through our acre, we are within its **zone of influence**; bears can see and hear the traffic go by. The highway belongs to a certain **disturbance class** (the linear, motorized, high intensity disturbance class) which may be presumed to reduce habitat effectiveness in covered vegetative areas by three-quarters during the summer season. That is, US 89 has a **disturbance coefficient** of 0.25. Disturbance coefficients range from zero to one; a DC of 1.0 has no impact on habitat effectiveness, a DC of 0.33 cuts it in third, and a DC of 0 reduces it to nil.

In the absence of any disturbances, the habitat effectiveness of an area equals its habitat value. US 89 reduces the habitat effectiveness of our acre to a fourth its previous value:

$$HE_{July\ 4} = HV_{July\ 4} \times DC_{US\ 89\ on\ July\ 4}$$

$$HE_{July\ 4} = 2.0 \times 0.25 = 0.5$$

What about the campground? The CEM considers the effects of multiple, overlapping activities to be (surprise!) cumulative. Assuming the campground belongs to the "point, motorized, high intensity" disturbance class with a July 4 disturbance coefficient of 0.5 for covered locations, the final habitat effectiveness for our acre is:

$$HE_{July\ 4} = HV_{July\ 4} \times DC_{US\ 89\ on\ July\ 4} \times DC_{Campground\ on\ July\ 4}$$

$$HE_{July\ 4} = 2.0 \times 0.25 \times 0.5 = 0.25$$

There may be many kinds of human activities occurring on the landscape at continuously variable levels of use and intensity. The ICE Tool Kit allows activities to have different disturbance classes from one season to the next (or from one day to the next) to reflect changes in activity use patterns throughout the year.

Disturbance classes have traditionally been based upon activity form (point, linear, or dispersed), whether motorized or non-motorized, and level of use (high, low, or incidental). ICE, however, allows you to split or lump disturbances into any number of classes reflecting biological considerations within your ecosystem.

Each disturbance class has a set of disturbance coefficients, one for each security cover class. In our example, each disturbance class has one DC for covered areas, and another DC for open areas.

Each disturbance class also has a zone of influence radius and a **decay function.** The decay function determines how the disturbance coefficient "decays" with increasing distance from the activity. ICE offers you the choice of three decay functions; constant over distance, inversely proportional to distance from the activity, and inversely proportional to the square of the distance from the activity.

Mortality Risk Index

The CEM considers the presence of activities to increase the mortality risk of grizzly bears. Towards this end, activities are not only identified by their disturbance class, but also by their **mortality class.** Just as for disturbance classes, each mortality class has a zone of influence, decay function, and set of **mortality coefficients** (MC).

Let's assume US 89 has a "firearms present, attractants absent" mortality class with a mortality coefficient of .0001 on July 4, and the campground has a "firearms present, attractants present" mortality class with an MC of .0005. The **mortality risk index** (MRI) for the acre on July 4 is:

$$MRI_{July\ 4} = MC_{US\ 89\ on\ July\ 4} + MC_{Campground\ on\ July\ 4}$$

$$MRI_{July\ 4} = 0.0001 + 0.0005 = 0.0006$$

If we have sufficient mortality data to state, for example, that major highways account for .0001 grizzly bear deaths per acre of zone of influence per day per 10,000 bears, the MRI is a reasonable estimator of bear mortality associated with each activity. On the other hand, if our mortality coefficients are more qualitative, the MRI is an open ended index useful in comparative analysis only (an MRI of 20 is twice as risky as an MRI of 10).

Some ecosystems include the habitat value in the MRI calculation:

$$MRI_{July\ 4} = HV_{July\ 4} \times (MC_{US\ 89\ on\ July\ 4} + MC_{Campground\ on\ July\ 4})$$

$$MRI_{July\ 4} = 2.0 \times (0.0001 + 0.0005) = 0.0012$$

Summing It Up

Vegetative, edge, and supplementary coefficients ... security cover classes, edge buffer zones, and edge classes ... disturbance and mortality classes, coefficients, zones of influence, and decay functions ... seasons and periods; these are all CEM **parameters** that vary between (and occasionally within) grizzly bear **ecosystems.** It is the structure of the CEM equations and ICE software that gives the CEM process consistency between ecosystems. It is the parameters that tailor the process inputs and outputs to your specific ecosystem.

Grizzly bear ecosystems are large islands of grizzly bear habitat covering millions of acres. These are divided into **bear management units** (BMUs) along topographic and biological boundaries of approximately 100,000 to 500,000 acres (actual sizes vary between ecosystems). BMU boundaries are drawn to encompass traditional bear concentrations and home ranges, and contain sufficient area and habitat mixture to support at least a sow with cubs.

BMUs are further divided into **subunits** of 50,000 to 250,000 acres. Subunit boundaries delineate variability within the BMU and may reflect, for example, changes in bear use patterns between seasons. ICE may be run at the subunit level to facilitate handling of the volumes of required data. Subunit results are additive into BMUs, and BMU outputs are additive up to the ecosystem level.

An ICE project may span one or more subunits or bear management units.

Let's return to our little acre example where we derived the HV, HE, and MRI for one day, July 4. This is just one of hundreds of thousands of acres making up the bear management unit. The HV, HE, and MRI for the entire BMU on July 4 is simply the sum of those values for all its individual acres. The per acre average is obtained by dividing its BMU sum by the BMU acres.

Similarly, the HV, HE, and MRI for the BMU within a specific season or **period** is simply the sum of the HV, HE, and MRIs for all days in the period. The seasonal daily averages are obtained by dividing the seasonal sums by the number of days in the period.

Finally, if you like to work with smaller numbers, you can divide the area's seasonal sums by both the BMU acres and the number of days in the period, yielding the average HV, HE, or MRI per day per acre for the period.

A CEM analysis project is divided into a gridwork of **cells** (figure 1). The ICE Tool Kit uses the methods we just covered to determine HV, HE, and MRI at the cell level. While we used an acre as our example cell size, ICE allows you to specify the resolution for each project.

And that, folks, is basically CEM in a nutshell. Go have a cup of coffee.

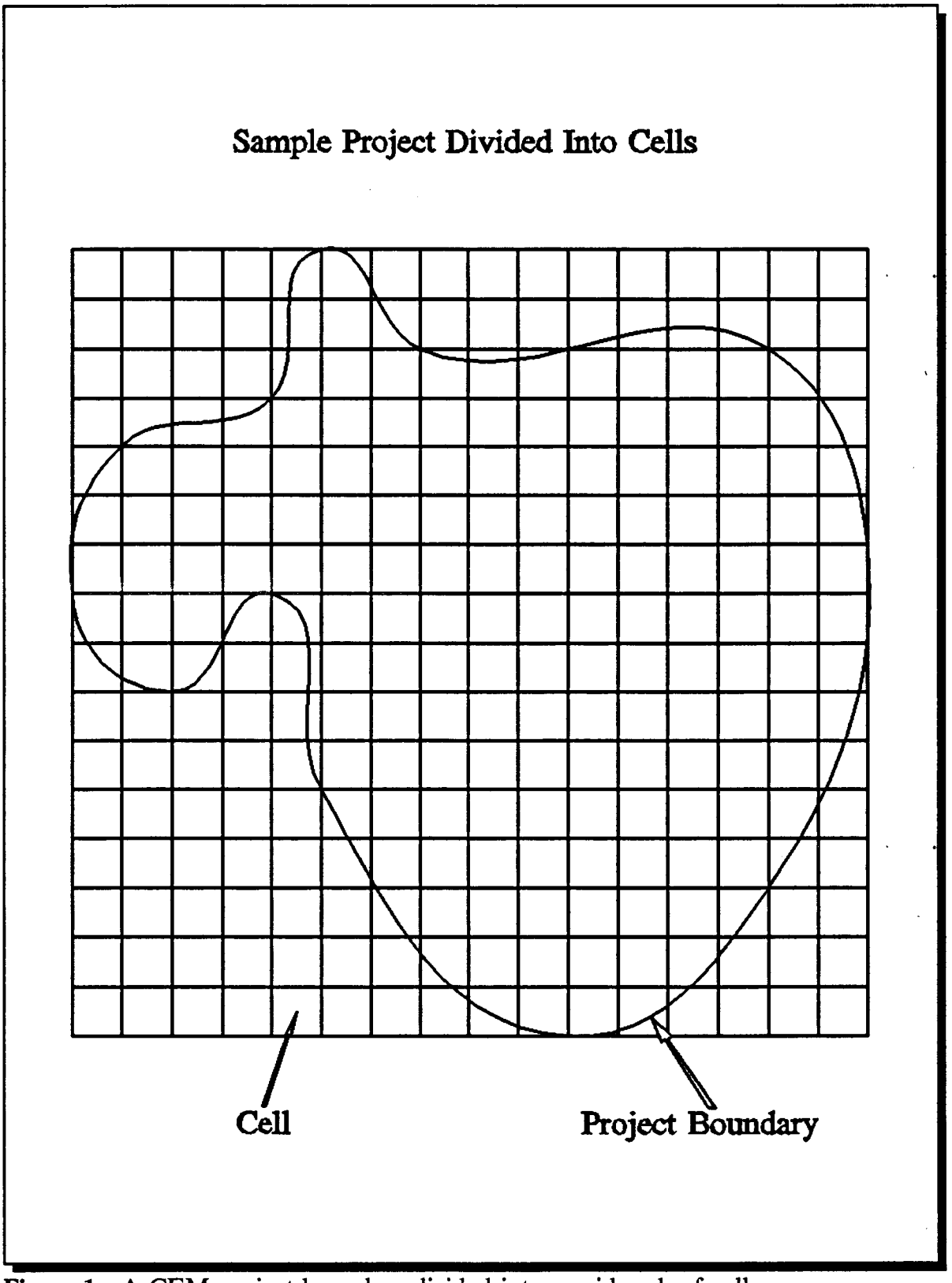

Figure 1 - A CEM project boundary divided into a gridwork of cells.

1.2 The ICE Tool Kit Metaphor

The structure of a system reflects the structure of the organization that built it.
Richard E. Fairley
Wang Institute

It seems like all software packages these days are based upon some metaphor or another; there are electronic spreadsheets, tabular data bases, computerized note pads, contact sheets, time cards, index cards, telephone directories, and even entire desk top metaphors complete with garbage cans.

In an effort to remain contemporary, ICE Tool Kit uses the map table metaphor. Please envision a large map table built upon a map file cabinet, the kind with a dozen two-inch deep drawers. The table has a blank, gridded surface with one or more one-year calendars setting on it.

You create a separate cabinet-table-schedule for every project using an ICE Tool Kit program, PROJNEW. The program builds the map and data drawers, etches the grid work on the table top, and sets a blank calendar on top. You then load the project parameters into the metaphorical data drawer.

In our metaphor, maps are created on sheets of clear plastic of any size. One or more **features** belonging to a single **theme** are painted on the map in opaque ink. A vegetative map, for example, could have several "Douglas-fir" features and several "whitebark pine" features. The maps may be completely, or only partially, painted with features. It may take several map sheets or quadrangles to complete a single **coverage** with a specific theme vintage (such as 1980 vegetation) for a project.

The ICE Tool Kit recognizes three habitat themes and three activity themes:

- **Unit boundary** maps identify boundaries within the project area. Unit boundaries may be tiled, overlapping, or nested.
- **Vegetative** maps identify various vegetation and security cover features. The vegetation coverage, in turn, determines the **security cover** and **edge** coverage.
- **Supplemental habitat** maps identify various supplemental habitat features.
- **Point activity** maps identify the locations of point activity features.
- **Linear activity** maps identify the locations of linear activity features.
- **Dispersed activity** maps identify the locations of polygon activity features.

The vegetative and supplementary habitat map features are color coded (metaphorically) to **attribute classes**; each attribute class has a calendar of vegetative or supplementary coefficient values. A whitebark pine vegetative feature may be color-coded to the "Highly Preferred" attribute class.

The activity maps are accompanied by an **activity attribute file** specifying the disturbance and mortality classes by date for each activity.

When completed, the maps and features are labelled and stored in the map cabinet. The ICE Tool Kit has one program, HABSTORE, for storing unit boundary, vegetative, and supplemental themes, and another program, ACTSTORE, for storing the activity maps and schedules. The programs validate the map and schedule data, build activity zones of influence, rasterize the map features, and store the features and activity schedules in our map cabinet (NOT on the map table).

When a unit boundary map containing a new unit is first stored in the cabinet, a clean one-year calendar is set upon the map table to hold its HV, HE, and MRI values.

Now you begin the process of compiling your baseline or current situation map from the map sheets stored in the cabinet. An ICE program, SHUFFLE, lets you extract one or more maps or features from the cabinet and lay them at a specific **position** or layer upon the map table. Maps and features at position 1 sit directly on top of the table, while maps at position 2 sit above position 1 maps, and so forth.

Let's assume you have a complete coverage of 1980 vegetative maps for the project. You also have a coverage of actual harvest areas from 1980 through 1990, and a coverage containing proposed harvests for 1991 through 1995.

You would first SHUFFLE the 1980 vegetation maps from the cabinet to position 1 on the map table. Next, SHUFFLE the 1980-1990 harvest maps to position 2. When you now look down on the map, you will see all the position 2 features and any position 1 features not overlaid by the harvest units; this represents your current situation. Finally, SHUFFLE the proposed harvests to position 3 to produce the proposed alternative.

SHUFFLE is used to move maps (1) from the cabinet to the map table, (2) from the map table back into the cabinet, and (3) between positions on the map table.

While you've been shuffling maps around, ICE has automatically been keeping track of incremental changes in the habitat value, habitat effectiveness, and mortality risk index for every unit in the project. You need only look at the CURRENT calendar for each unit to see how HV, HE, and MRI are being affected throughout the year.

Where SHUFFLE allows you move habitat and activity maps around, another program, SCHEDULE, allows to you change the disturbance and mortality classes and dates for selected activities. Once again, as you experiment with activity schedules, ICE determines the incremental effects on the current HV, HE, and MRI for all units in the project.

Once you have composed a set of coverages, maps, and features on the map table to represent a specific condition, ICE provides a software camera called SNAPSHOT to preserve the precious memory in living black and white. SNAPSHOT simply makes a copy of the unit's CURRENT HV-HE-MRI calendar and stores it with a label and description of your choosing (such as "BASELINE 1990" or "1995 Alternative 1").

Finally, the ICE Tool Kit provides you with a couple output summary programs. COMPARE creates a table comparing HV, HE, and MRI values by season for two calendars in a project unit. It would, for example, show the absolute and percent difference between the "BASELINE 1990" and "1995 Alternative 1" calendars.

The BARCHART program, available on MS-DOS computers with Quattro Pro, selects the HV, HE, or MRI values for a season from unit calendars you specify, puts them into a Quattro Pro spreadsheet, starts up Quattro Pro, and creates a bar chart of the results, all without your intervention. When control is returned to you, you are inside Quattro Pro with a bar chart on your terminal. You may then edit, save, and print the chart.

While ICE includes several other small support and maintenance programs, this is the meat of the system. You create and store maps, shuffle them around on the map table, experiment with activity schedules, take snapshots, and compare and display the snapshots.

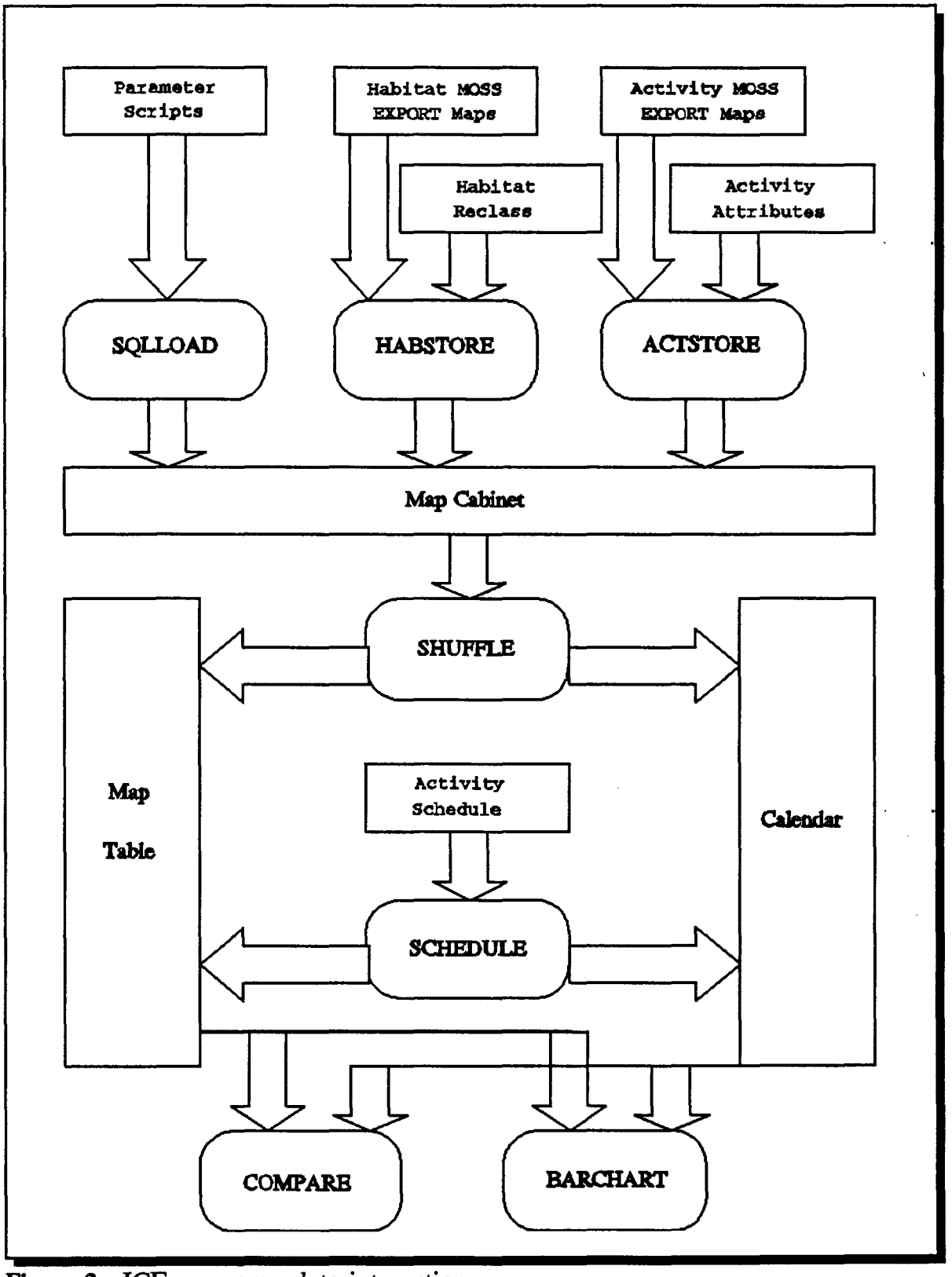

Figure 2 - ICE program - data interaction.

1.3 CEM Steps

Below is a list of steps required to complete a CEM cycle for a project. Whether you begin at step 1, 2, 3, 4, or 5, depends upon whether you are the first in your ecosystem or at your computer installation to use the ICE Tool Kit. While the steps are listed in the general order you begin them, there will be considerable overlap and cycling.

Data Preparation Phase

1 Acquire computer hardware and install the operating system, ORACLE relational data base management system, text editor, GIS, and ICE Tool Kit software (allow 2 or 3 days).

2 Run ICICLE to create the ICE_PROJECT Table in the data base (minutes). ICICLE is run just once on each computer.

3 Determine the ecosystem parameters and store them in SQL*LOADER script files (hours). These may already have been compiled for your ecosystem.

4 Create the ecosystem habitat reclass files for unit boundary, vegetation, and supplemental themes (hours). These may already have been compiled for your ecosystem.

5 Create the project's MOSS EXPORT map files from digitized maps, satellite images, or other GIS sources (weeks or months). You may need to use conversion utilities supplied with your GIS to create map files in MOSS EXPORT format. This is the most time-consuming step and the easiest to under-estimate!

6 Create activity attribute files for the project's point, linear, and dispersed activity maps (days or weeks).

Map Composition Phase

7 Run PROJNEW to define and create the new CEM project (minutes).

8 Run SQL*LOADER to load the ecosystem parameters (from step 3) into the project's data tables. (minutes).

9 Run HABSTORE to store the unit boundary, vegetation, and supplemental theme maps (from steps 4 and 5) in the project's map cabinet (minutes to hours).

10 Run ACTSTORE to store the activity maps and schedules (from steps 5 and 6) in the project's map cabinet (minutes to hours).

11 Run SHUFFLE to move maps from the cabinet onto the map table, composing a specific habitat-activity condition (minutes to hours). The impacts on HV, HE, and MRI values are automatically recorded on each unit's CURRENT calendar.

12 Run SCHEDULE to modify activity schedules and disturbance-mortality classes (minutes to hours). The impacts on the HV, HE, and MRI values are automatically recorded on each unit's CURRENT calendar.

13 Run SNAPSHOT to preserve the HV, HE, and MRI calendars of the project's current map table composition (minutes).

14 Repeat steps 9 through 13 as necessary to compose various management alternatives (the rest of your life).

Results & Analysis Phase

15 Run COMPARE to produce tables comparing HV, HE, and MRI values by period between project unit calendars (e.g., between alternatives) (minutes).

16 Run BARCHART to produce bar charts comparing the HV, HE, and/or MRI for a period (or all periods) between project unit calendars (minutes).

17 Draw conclusions and make recommendations (???).

Shut Down Phase

18 Run ORACLE's EXP utility to backup project data tables (minutes to hours).

19 Run PROJDROP to remove the project from the data base (minutes).

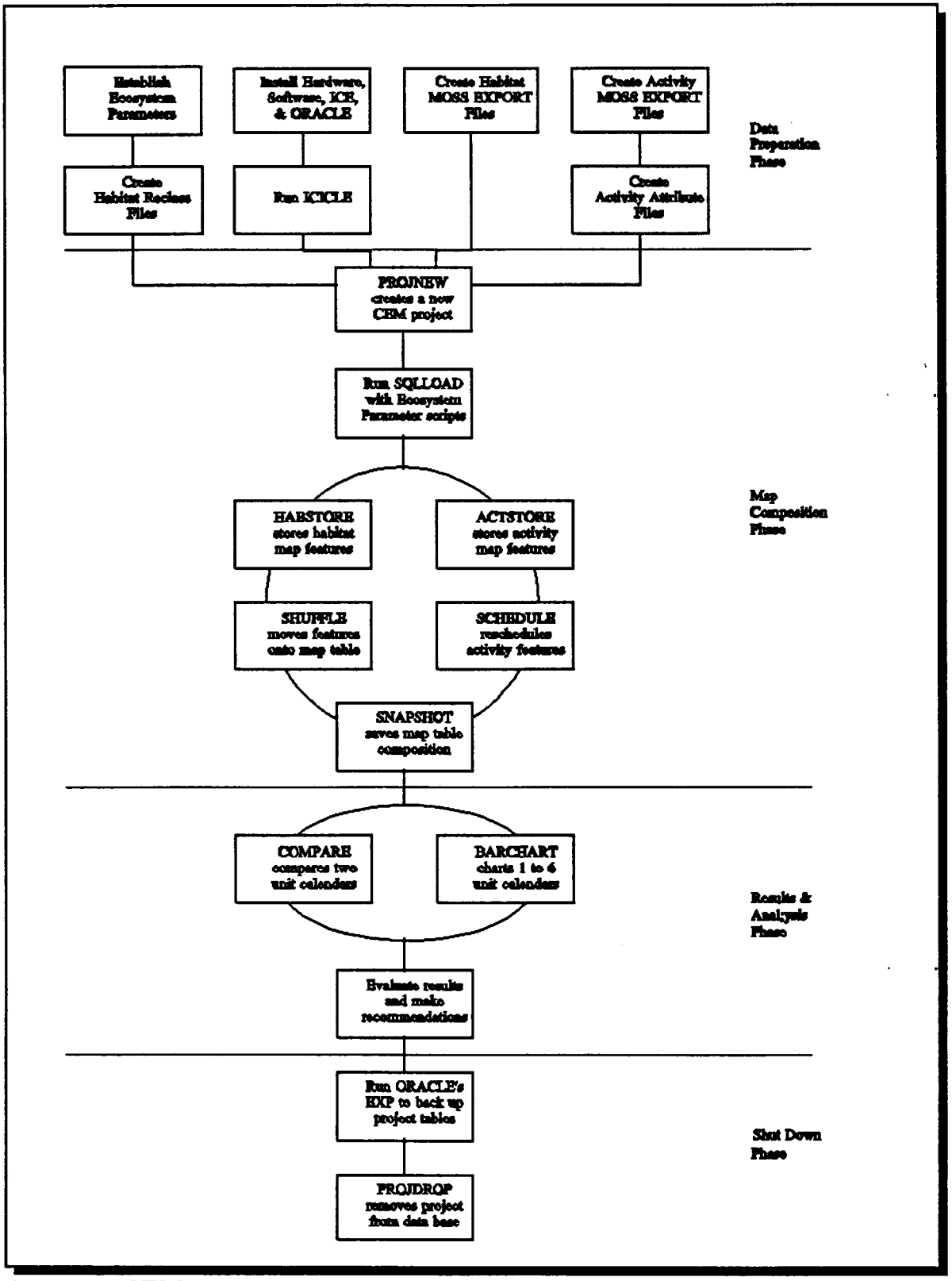
Establish Ecosystem Parameters
Install Hardware, Software, ICE, & ORACLE
Create Habitat MOSS EXPORT Files
Create Activity MOSS EXPORT Files
Data Preparation Phase
Create Habitat Reclass Files
Run ICICLE
Create Activity Attribute Files
PROJNEW creates a new CEM project
Run SQLLOAD with Ecosystem Parameter scripts
HABSTORE stores habitat map features
ACTSTORE stores activity map features
Map Composition Phase
SHUFFLE moves features onto map table
SCHEDULE reschedules activity features
SNAPSHOT saves map table composition
COMPARE compares two unit calendars
BARCHART charts 1 to 6 unit calendars
Results & Analysis Phase
Evaluate results and make recommendations
Run ORACLE's EXP to back up project tables
Shut Down Phase
PROJDROP removes project from data base

Figure 3 - CEM steps.

1.4 Learning Curves

The list of CEM tasks should give you some idea of the work that lies ahead. Be prepared to scale learning curves for:

- your computer's operating system (directory structures, pathways, copying files, backing up, etc.),
- your GIS system (digitizing, displaying, converting file formats, etc.),
- your text editor,
- the ORACLE SQL*PLUS query language, and
- the ICE Tool Kit concepts, organization, and programs.

1.5 Scale of Use

When all you have is a hammer, every problem looks like a nail.
Anonymous

The ICE Tool Kit determines the cumulative effects of activities on bear habitat by accumulating the incremental impacts of single activities on the habitat. The incremental impact of shuffling or rescheduling an activity is reported as output, and may be used to play out mitigation scenarios. The appropriateness of doing so, however, depends entirely upon the resolution, accuracy, and precision of the model parameters and map data.

If we understand the relationship between bear use patterns, habitat, and activity disturbance sufficiently well, and are able to accurately map an area at sufficient detail (e.g., with resolution in the 1 to 10 meter range), then ICE may be an appropriate tool for performing analysis at the project level; we could game the impact of a specific, known activity located at a specific, mapped location.

The target of cumulative effects modeling has historically centered on entire bear populations within an ecosystem. Our knowledge of the relationships between bear use patterns and mortality with habitat features and human activity is generally qualitative; certain classes of activities generally result in certain classes of impacts when occurring within certain classes of habitat.

When we evaluate the effects of a specific activity at a specific location using these generic classes, a certain amount of error results. We may expect this error to be relatively large compared to the activity's incremental effects, since the classes reflect an ecosystem scale. Over the entire ecosystem, however, the errors should average out to some smaller, acceptable value relative to the ecosystem scale.

As a result, our ecosystem-level parameters and data are probably not sufficiently accurate for playing out mitigation scenarios on an activity-by-activity basis. The data resolution and accuracy should be sufficient, however, for evaluating the impacts of systems of activities on the ecosystem.

Avoid using a sledge hammer to repair a Swiss watch.

1.6 Review

- A **project** is a rectangular area (the map table top) containing one or more **units.** Units may be **bear management units, subunits,** drainages, home ranges, or areas of special interest. You may have any mix of units within a project; continuous, discontinuous, tiled, overlapping, and nested.

- A **theme** describes the unit boundaries, vegetation, supplemental habitat, point, linear, or polygon activities. A **coverage** is an instance of a theme covering the entire project (such as 1980 vegetation). The coverage may be composed from one or more **maps** containing one or more **features.**

- **Unit boundary, vegetation,** and **supplemental habitat** features are polygons with a specific attribute. The attribute specifies the unit, vegetation class, or supplemental habitat class to which a feature belongs.

- **Activity features** are points, lines, and polygons surrounded by a **zone of influence.** Each activity has a **schedule** specifying its **disturbance** and **mortality** class between a set of dates. An activity may have up to 365 dates in its schedule.

- The vegetation coverage determines the **security cover** and **edge** coverages. These coverages, combined with the supplemental habitat coverage, determine the **habitat value** for each **period** of the year.

- The activity zone of influence coverage and schedule determine the **habitat effectiveness** and **mortality risk index** for each period of the year.

- The habitat value, habitat effectiveness, and mortality risk index values are automatically calculated as you reschedule activities and shuffle maps on the **map table.** The results are stored in the **CURRENT calendar** for each unit in the project.

- You may take a **snapshot** of any CURRENT calendar to record the existing map table conditions. Snapshots of various alternatives may be compared and analyzed later.

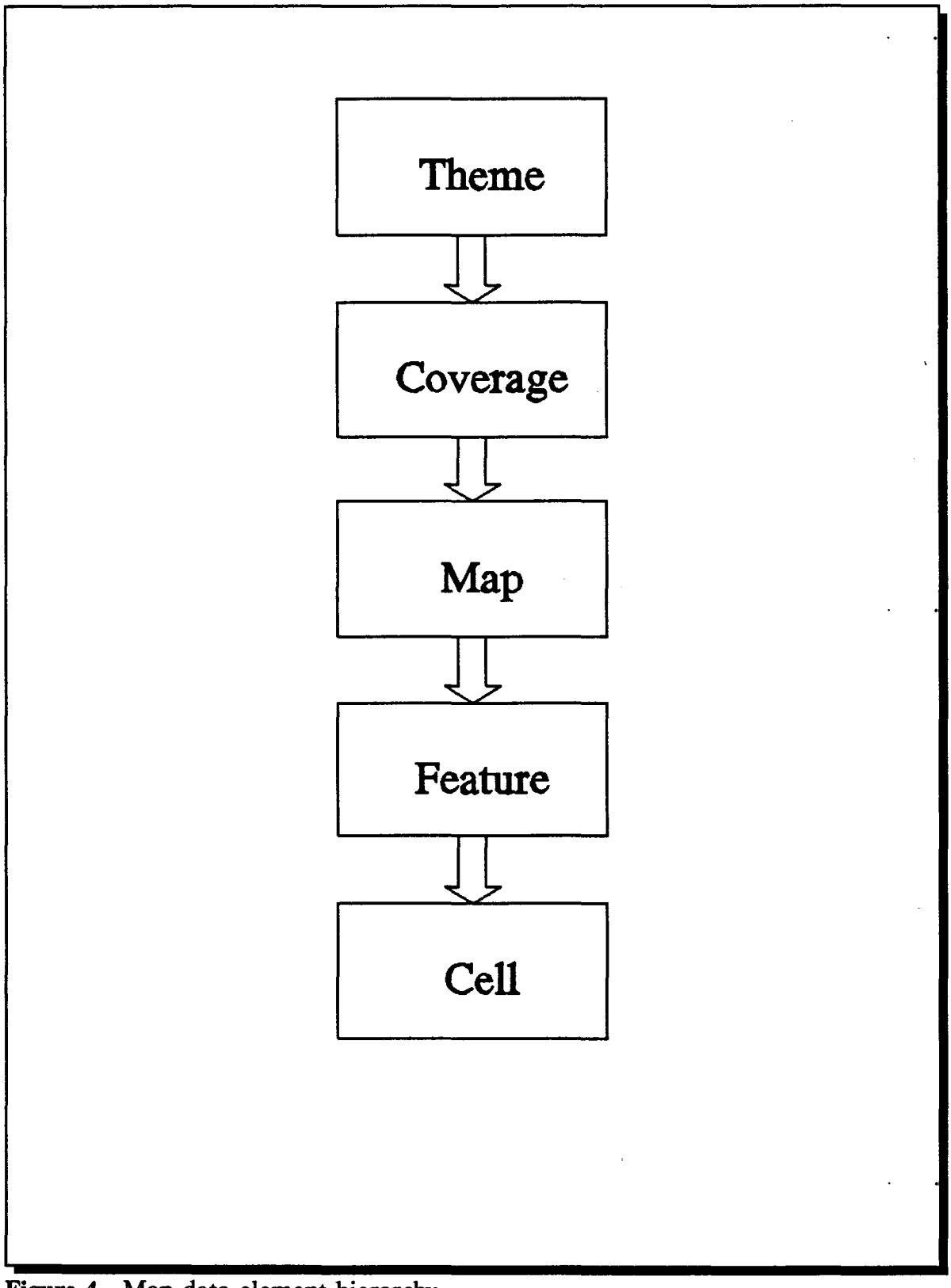

Figure 4 - Map data element hierarchy.

1.7 What's Next?

Section 2 of this manual addresses the data preparation tasks, which will consume about 90% of your allotted time (the remaining tasks will consume the other 90% of your allotted time).

Section 3 introduces a very small sample project and illustrates how it is processed through the ICE Tool Kit programs.

Section 4 documents all CEM analysis and support programs contained in the ICE Tool Kit.

Section 5 documents the formats of the various ICE data files.

Section 6 documents the ORACLE table structures used by ICE. You will reference this section when you interactively query the data base via SQL*PLUS.

The appendices include information on hardware and software installation, ecosystem-specific programs and parameters, and other technical notes.

Enjoy!

2. DATA PREPARATION

Data preparation consumes the greatest portion of blood, sweat, and tears during an incremental cumulative effects analysis.

First, you must complete the obvious task of acquiring and setting up a computer system to support CEM. You should allow 2 or 3 days to set up and test your computer hardware and install and test the software. You will have to install the operating system, partition the hard disk, then install the text editor, ORACLE, the ICE programs, and any additional software (communications programs, disk utilities, etc.).

In addition to the time you must spend learning the ICE Tool Kit, you may also have to spend several days learning the MS-DOS operating system and the text editor.

Ecosystem parameters such as habitat, disturbance, and mortality classes and their coefficients, must be determined and entered into **SQL*LOADER control files.** Fortunately, this process needs to be completed only once for an ecosystem and may already be available to you.

The biggest task you face is building the map files. Map data may be digitized from existing or modified maps and orthoquads, or obtained from satellite imagery. Both methods require classification and validation. The map data must be converted to MOSS EXPORT format for subsequent storage into an ICE project map cabinet.

Unit boundaries, vegetation, and supplemental habitat maps usually require ecosystem-specific **habitat reclass files.** Oftentimes the map features are classified by one scheme, such as habitat-cover type code or spectral classification code, and the CEM recognizes another scheme, such as grizzly bear preference class. The reclass files are concordances specifying the cross-walk between the two classification systems. Once again, these three files (one each for unit boundary, vegetation, and supplemental habitat) may already exist for your ecosystem.

Point, linear, and dispersed activity maps dictate where activities occur on the landscape. The **activity attribute file** that accompanies an activity map identifies each activity and specifies when it occurs. An activity may occur any number of times during the year, each time with a different disturbance-mortality class combination.

When you have completed this litany of tasks, you are ready to run the ICE Tool Kit.

2.1 Step 1: Hardware & Software Installation

The following hardware should be installed for this release of the ICE Tool Kit:

- computer with an Intel i80486 or i80386/i80387 CPU (required),
- 2 megabytes system memory required (16 megabytes recommended for disk caching),
- hard drive(s) with 100-300 megabytes for general system use and 500 megabytes for CEM and ORACLE use,
- monitor (color VGA recommended),
- keyboard (enhanced 101-key recommended),
- magnetic tape drive (highly recommended),
- 3.5" and 5.25" floppy drives (1 required, both highly recommended),
- 2 serial and 1 parallel port (highly recommended),
- mouse (recommended),
- 2400 baud MNP/5 modem (recommended).

You must also have the following software installed on the computer:

- MS-DOS 3.3 or later (required),
- ORACLE Professional Version 5.1C (required, Version 6.0 not acceptable),
- QEMM memory manager (required),
- ICE Tool Kit software (required),
- program/text editor capable of handling very large files with very long lines (such as Sage Professional Editor or VEDIT),
- disk caching software such as PC-KWIK (highly recommended),
- Quattro Pro spreadsheet software for use with BARCHART program,
- backup software for the tape drive (highly recommended).
- communications software for the modem.

CEM will consume large tracts of hard disk space, so it is most important that you have sufficient hard disk memory. Second, having lots of extra system memory (RAM) to serve as a disk cache will greatly speed program performance.

Note that CEM data and program requirements may force you to dedicate the machine to CEM use.

2.2 Step 2: Run ICICLE to Create ICE_PROJECT Table

Once your hardware is set up and you have the ORACLE rdbms installed, you must run the ICICLE program to create the ICE_PROJECT Table. This table will contain all the ICE project definitions as you create them.

Just as ORACLE is only installed once on your machine, ICICLE is only run once on your machine. It takes only a few seconds to run. Please refer to the ICICLE program reference in Section 4.

2.3 Step 3: Ecosystem Parameters

One person's constant is another person's parameter.
Susan Gerhart
Microelectronics and Computer Technology Corporation.

While the ICE Tool Kit provides a common analytical structure to the cumulative effects analysis process, the ecosystem parameters tailor the analysis to the ecosystem. Ecosystem parameters are stored in nine data base tables.

The **Theme Table** contains the label, description, id number, and zone-of-influence for the cover, edge, unit boundary, vegetation, supplemental habitat, and point, linear, and dispersed activity themes.

The **Period Table** contains the begin date, end date, label, and description for each period or season. The ecosystem may have 1 to 365 periods.

The **Displacement Table** contains the label, description, id number, zone-of-influence, and decay function for each activity disturbance class recognized within the ecosystem. The **Displacement Cover Table** contains the source and peripheral disturbance coefficients for each security cover class of each disturbance class.

The **Mortality** and **Mortality Cover Tables** contain information analogous to the Displacement and Displacement-Cover Tables.

The **Attribute Table** contains the label, description, id number, and cross-reference number for the security cover, edge, unit boundary, vegetation, and supplemental habitat classes. The **Attribute Date Table** contains the class coefficients by date. Each class may have from 1 to 365 coefficients with their begin and end dates.

Finally, the **Edge Transition Table** is a matrix defining edge conditions. The matrix rows identify source cell security cover type, the columns identify neighboring cell security cover type, and their intersection specifies the resultant source cell edge type.

The nine tables are created when you run the ICE PROJNEW program to define a new CEM project. A separate set of empty tables are created for each project.

To fill the tables, you must run the ORACLE SQL*LOADER program SQLLOAD once for each table. SQLLOAD reads a control file containing the data to store in a single table. This only takes a few minutes, especially if run from a batch file. The hard part is building the control files in the first place.

Fortunately, the control files may already exist for your ecosystem (check the ICE Tool Kit distribution diskettes). If so, you must edit the files before running them. If not, you must create the script files.

If SQLLOAD control files already exist for your ecosystem, you must edit them before running SQLLOAD for a new project. This simply involves changing the period label prefix to the table name (in the figure below, "SAMPLE_THEME" on line 12 must be modified to "<new_project>_THEME").

Section 5.8 documents control file structure and considerations you must make if creating them from scratch.

```
-- SQL*LOADER control file to load an empty Theme Table.
-- You must modify the table name in line 12 to match the project label.
-- Collin D. Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE THEME.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_THEME
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(THEME_NUMBER, THEME_LABEL, THEME_DESCRIPTION, THEME_ZONE, THEME_TOP_POSITION)
--
BEGINDATA
11, "COVER",      "Security Cover",                    0, 0
12, "EDGE",       "Habitat Edge",                    100, 0
21, "UNIT",       "Unit Boundary",                     0, 0
31, "VEGETATION", "Vegetation: Habitat & Cover Type",  0, 0
32, "SUPPLEMENT", "Supplemental Protein Source",       0, 0
41, "POINT",      "Point Activity",                 1000, 0
42, "LINEAR",     "Linear Activity",                1000, 0
43, "DISPERSED",  "Dispersed Activity",                0, 0
```

Figure 1 - Sample SQLLOAD control file to load a project Theme Table.

2.4 Step 4: Habitat Reclass Files

Oftentimes the habitat features delineated on your maps are not classified using the same scheme as the cumulative effects model. Vegetation polygons may have been labeled by habitat and cover type ("73453" for example), but your ecosystem model recognizes vegetation by grizzly bear preference (none, low, moderate, or high).

The reclass files specify how map subjects are reclassified into CEM attributes for the ecosystem. The ecosystem has one reclass file each for unit boundary, vegetation, and supplemental habitat map themes. The ICE HABSTORE program uses the appropriate file to translate map subjects into CEM attributes.

A reclass file is a simple ASCII text file created and modified with a text editor. In figure 2, the habitat-cover type "72044" is assigned vegetation class "V:X:H" (vegetation, extremely high preference, high security cover).

Habitat reclass files are fully documented in section 5.4.

```
ICE RECLASS
00011                    V:X:L
0001172044               V:X:LH
7204400011               V:X:HL
72044                    V:X:H
CLEARCUT                 V:L:R
NU                       NONE
"POND OR LAKE"           V:N:L
```

Figure 2 - Sample Project reclass file.

The ICE HABSTORE program reads and validates the reclass file before storing any map information. If the file contains errors such as nonexistent attribute labels, HABSTORE reports them to you and terminates.

Reclass files can become very long. The Greater Yellowstone Ecosystem, for example, mapped vegetation using habitat-cover type codes, of which there are several hundred. Furthermore, a polygon may be coded as a **mosaic** of two or even three types. Each occurrence of these single, double, or even triple codes must be represented in the vegetation reclass file. Otherwise HABSTORE will read the polygon subject from the MOSS EXPORT file, fail to find an appropriate attribute class to throw it in, display an error message, and terminate. You would then add that map subject to the reclass file and try again.

Reclass files for unit boundaries and supplemental habitats usually contain fewer than 50 records.

2.5 Map Preparation and MOSS EXPORT Files

Your original map data may be generated by a digitizer and/or obtained from a satellite image. Digitizers produce maps in **vector** format wherein a point is a single coordinate pair, a line is a series of points, and a polygon is a closed line (which may contain other polygons). Each point, line, or polygon is a **feature** that is labelled with a unique feature label ("FSR 1234") or with a non-unique class label ("LOW_PREFERENCE").

A satellite image, on the other hand, is in **raster** format; a rectangular grid of discrete pixels or cells. Each cell has a (non-unique) class label, usually a number. The cells are not explicitly identified with any map feature; features must be determined by identifying the boundaries of contiguous cells with the same class label.

The ICE Tool Kit keeps track of map information at the feature level. First of all, we tend to think of roads and harvest units as entire features rather than as collections of cells. It is more natural to request a road closure than a cell set closure. Second, it is more economical to store lineage and attribute information for several thousand features than it is for several hundreds of thousands of cells.

The ACTSTORE and HABSTORE programs read map files in MOSS EXPORT format and stores them in the project's map cabinet. This format was chosen because:

- the MOSS EXPORT format represents map data at the feature level,
- MOSS EXPORT files are simple ASCII text files that are easy to read and edit,
- the MOSS EXPORT format is supported either directly or indirectly by most popular geographic information systems (including ARC/INFO and ERDAS), and
- much of the CEM mapping effort to date has produced map files in this format.

The ARC/INFO GIS has a program (ARCMOSS) to convert a coverage into MOSS EXPORT format. It has another program (GRIDPOLY) that converts raster maps in ASCII, ERDAS 8-bit, ERDAS 16-bit, EPPL7, and MIADS format into ARC/INFO coverages, from whence they can be run through ARCMOSS.

MOSS EXPORT files are simple ASCII text files you can view and edit with a text editor and list on a printer. Figure 3 shows the MOSS EXPORT file for the Sample Project vegetation map.

Each feature (point, linear, or polygon) in the file begins with a **header record** and is followed by one or more **coordinate pair records.**

```
  1          72044                                    5
000000.00 0000500.00
000000.00 0001000.00
001000.00 0001000.00
001000.00 0000500.00
000000.00 0000500.00
  2          00011                                    5
000000.00 0000000.00
000000.00 0000500.00
001000.00 0000500.00
001000.00 0000000.00
000000.00 0000000.00
```

Figure 3 - Vegetation map MOSS EXPORT file for the Sample Project.

The header record contains the feature's **item number, subject,** and **number of points.** The item number is usually a sequence number assigned by the GIS, and may correspond to the original paper map from which it was digitized. The subject contains the unit boundary, vegetation, or supplemental habitat class label, or an activity feature label. The number of points field indicates the number of coordinate record pairs that follow.

The coordinate pair records contain an X and Y coordinate pair. These are usually UTM eastings and northings. Only a single record must occur for point features. Any number of records may occur for linear and polygon features.

MOSS EXPORT file formats are fully documented in section 5.5.

You will need to prepare MOSS EXPORT files for the three habitat themes (unit boundaries, vegetation, and supplemental habitat), and the three activity themes (point, linear, and dispersed activities). You may need several files per theme to create a coverage for the entire project area.

You may also have several coverages or vintages of a theme to represent, for example, the 1984 baseline vegetation survey, 1984-1990 harvest activity, 1984-1990 fire activity, and 1991-1995 proposed harvest activity.

2.6 Step 6: Activity Attribute Files

The activity map MOSS EXPORT files define where activities are located. The attributes of the activities and their schedules of occurrence are defined in the **activity attribute files.**

Every activity map MOSS EXPORT file must have a corresponding activity attribute file. And every feature in the MOSS EXPORT file must have a corresponding block of records in the activity attribute file.

The activity attribute file assigns a label and description to the activity as well as a beginning date, an ending date, and a disturbance and mortality class per period. An activity may have from 0 to 365 periods defined, allowing you great flexibility in describing activity levels throughout the year.

Figure 4 illustrates a sample linear activity attribute file. It is a standard ASCII text file with "ICE ACTIVITY ATTRIBUTES" entered on the first line. Lines beginning with an asterisk are comments. The file is a free-format; fields within a line are separated by one or more spaces or tabs.

```
ICE ACTIVITY ATTRIBUTES
*
* SAMPLE Linear Activity Attributes
*
ACTIVITY 'HIWAY 1:1' 'A' 'HIWAY 1 SEGMENT 1'
DATES 0101 1231 L_M_H L_H_F
ACCEPT
ACTIVITY 'HIWAY 1:2' 'A' 'HIWAY 1 SEGMENT 2'
DATES 0101 1231 L_M_H L_H_F
ACCEPTSAME
ACTIVITY 'SPUR 1:1' 'A' 'SPUR 1 SEGMENT 1'
DATES 0101 1231 L_M_L L_L_F
ACCEPT
```

Figure 4 - Sample activity attribute file.

An activity is defined by a block of lines consisting of:

- a single ACTIVITY record,
- zero or more DATES records, and
- a single ACCEPT-ACCEPTSAME-REJECT record.

The ACTIVITY record begins with the keyword "ACTIVITY". The second field on the line is a 1 to 16-character activity label. Labels often reference the activity's source map and feature (e.g., BIGSP123 for Big Springs quadrangle, activity 123). The third field is the activity group or alternative label. The fourth field on the line is a 0 to 64-character activity description.

The second line defining an activity is usually a DATES record. It begins with the keyword "DATES". The second and third fields are the beginning and ending dates, inclusive, in "mmdd" format. The fourth and fifth fields contain valid disturbance and mortality class labels. An activity may have 0 to 365 DATES records.

The final line in an activity record block contains one of the keywords ACCEPT, ACCEPTSAME, or REJECT. If REJECT is specified, the activity is not stored in the map cabinet.

If ACCEPT is specified, the activity is stored in the map cabinet at the next highest activity map position. Normally, each feature is treated as its own complete activity and occupies its own position on the map table. Activities that occur at different positions on the map table have cumulative impacts.

If ACCEPTSAME is specified, the feature is treated as a segment or part of the previous activity. This behavior is useful in two situations.

In the first case, an activity such as a major highway may have been digitized as a series of smaller highway segments. If treated as individual activities, these segments will have overlapping zones of influence at their nodes, resulting in a double impact. This is usually not the intended effect.

In the second case, you may have a set of closely related activities, such as a road serving a housing development, where you want the model to use only the worst case activity from the set, and not all the activities from the set.

An ACCEPT activity feature may be followed by any number of ACCEPTSAME activity segments. This implies that all activity features belonging to the same activity set must appear contiguously in both the MOSS EXPORT map file and the activity attribute file.

See section 5.1 for a more complete discussion of activity attribute file formats and use of ACCEPT and ACCEPTSAME.

The ICE ACTSTORE program reads and validates both the MOSS EXPORT activity map and its attribute file before storing any information in the map cabinet. The program makes sure the MOSS EXPORT file headers are in the correct format, that the correct number of coordinate pair records follow, and that polygons are closed.

The disturbance and mortality class labels in the activity attribute file are also validated. There must be a one-to-one correspondence between activity features in the MOSS EXPORT file and activity features in the activity attribute file.

4. PROGRAM REFERENCE

The programs distributed with the ICE Tool Kit fall into three categories:

- *ICE Tool Kit Programs* read and/or write data to the data tables in the ORACLE relational data base. These programs require a command input file which specifies, among other things, your ORACLE user name and password.

- *ICE support programs* do not access the ORACLE data base. They are provided to, for example, validate and list MOSS EXPORT files. These programs take all their inputs from the command line. Output is usually to the standard output device (your console), and may be redirected into files. These programs are useful to users from all ecosystems.

- *Ecosystem support programs* also do not access the ORACLE data base. These programs are created on an as-needed basis to solve ecosystem specific problems. An example is a program to convert activity attributes files from an old format or an ORACLE table ASCII listing to the ICE activity attribute file format.

This section documents ICE Tool Kit and ICE support programs only. Documentation for ecosystem support programs may be available in the appendices.

4.1 ICE Tool Kit Program Reference

Starting ICE Tool Kit Programs

All ICE Tool Kit and support programs are command driven. To start a program, you type its name at the operating system prompt. If you then press the <Enter> or <Return> key, a screen full of information about the program is displayed. For example, to get a description of the PROJNEW command, type:

```
D:\> PROJNEW
```

and press the <Enter> key and the display in figure 1 is printed to your console. Note that the "D:\>" portion of the example is the operating system *prompt* displayed on your console; you type only the portion in **bold face type.**

To actually run an ICE Tool Kit program, you must supply the name of a command input file. This file contains all the information required by a program to successfully run. The content of these files vary between programs and are described in this section.

Let's assume you have created a PROJNEW command input file named PROJNEW.INP using your text editor. To start the program, type:

```
D:\> PROJNEW PROJNEW.INP
```

press the <Enter> key, and the program begins executing. All program output is displayed at your console.

You may instruct the program to record all its console output in a *log file.* You do so by specifying the log file name after the command input file name. To duplicate PROJNEW's output in a file named PROJNEW.LOG, type:

```
D:\> PROJNEW PROJNEW.INP PROJNEW.LOG
```

If the file PROJNEW.LOG does not exist, it is created. If PROJNEW.LOG already exists, it will be appended.

```
------------------------------------------------------------------------------
ICE PROJNEW Project Creation Program   Version 0.80  Apr 91
Copyright (c) 1991 by Systems for Environmental Management
------------------------------------------------------------------------------

USAGE:
------
PROJNEW   <input_file>  [<log_file>]
where     <input_file> contains the program input commands, and
          <log_file>   is an optional log file to capture output.

DESCRIPTION:
------------
PROJNEW allows you to create a new ICE project.
You define the project area in the <input_file>, and PROJNEW stores
the definitions and creates the necessary data base tables.
NOTE: After creating the new project, you should run SQLLOAD to load
project parameters into the newly created tables!!

INPUTS:
-------
User             Req  Data base user name
PAssword         Req  Data base user password
PROJect          Req  Project label (1-8 chars)
Description      Opt  Project description (0-64 chars)
ECOsystem        Opt  Ecosystem label (0-16 chars)
SPecies          Opt  Species label (0-16 chars)
CEllmap          Req  Cell map file (full pathname) (1-80 chars)
INItial          Opt  Initial slots per cell (default=32)
INCrement        Opt  Cell map slot increment size (default=8)
Zone             Req  UTM Zone number
WEst             Req  Project's western-most boundary UTM
SOuth            Req  Project's southern-most boundary UTM
EAst             Req  Project's eastern-most boundary UTM
North            Req  Project's northern-most boundary UTM
WIDth            Req  Cell width (meters)
Height           Req  Cell height (meters)
Mri*hv           Opt  Multiply HV into MRI value (YES or NO)
SUpplement       Opt  Supplement habitat valuation (CUMULATIVE, MAXIMUM, or TOP)

------------------------------------------------------------------------------
```

Figure 1 - All ICE programs display a screen of information when only the program name is entered on the command line.

ICE Command Input File Format

ICE command input files are ASCII text files you create and modify with a text editor. A command file may contain:

- blank records,
- comment records, which must begin with an asterisk, and
- command records.

Blank and comment records should be used liberally to document the purpose of each input file and to improve its readability. Comment records must have an asterisk as its first non-whitespace character.

Command records always contain two fields.

- Field 1 is an input command, usually a single word. Each program has its own list of valid commands. Some commands are required and others are optional.

- Field 2 is the command argument. The two fields must be separated by one or more blanks and/or tabs. If the second field contains embedded blanks or special characters, it must be enclosed in single or double quotes.

Command Syntax Conventions

This documentation uses the following conventions when describing command syntax.

- The command name is shown in upper case PRESTIGE ELITE font.

- Command arguments within angle brackets, < >, indicate the type of argument to specify. For example, the syntax "USER <oracle_user_name>" means you should enter your actual ORACLE user name as the argument, such as "USER SYSTEM".

- Command arguments within curly braces, {}, and separated by a vertical bar, |, are alternate choices; you may enter any one of the choices as the command argument. For example, "PROGRESS { NO | YES }" means you may enter either NO or YES as the command argument.

Universally Required ICE Commands

Three commands are required by all ICE Tool Kit programs:

- `USER < oracle_user_name >`
- `PASSWORD < oracle_password >`
- `PROJECT < project_label >`

If one of these are missing, an error message is displayed and the program terminates. (Actually, I lied; the PROJLIST program does not require the PROJECT command, since it lists all projects in the data base).

Universally Optional ICE Commands

Three commands may optionally be included in any ICE Tool Kit program command input file:

- `BELL { YES | NO }`
- `ECHO { YES | NO }`
- `PROGRESS { YES | NO }`

If the BELL command argument is "YES", the console bell rings when the program terminates.

If the ECHO command argument is "YES", the command input file is echoed to the console (and the log file) as it is read and processed.

If the PROGRESS command argument is "YES", intermediate results are displayed to the console (and log file) to assure you that the program is really running and just how far it has progressed. This option can significantly increase the size of the log files!

Universal Program Behavior

All ICE Tool Kit programs exhibit the following behavior:

- If invoked with just the program name, a screen of program information is displayed and the program terminates. The display includes a brief program description, invocation options, and command input file options.
- If the program is invoked with a single argument, the argument is presumed to be the command input file.
- If the program is invoked with two arguments, the first is presumed to be the command input file and the second is presumed to be a log output file.
- The program begins by displaying its name, version, copyright notice, and the start date and time.
- The program then attempts to open the command input file, and if specified, the log file. If unsuccessful, an error message is displayed and the program terminates with a non-zero status code.
- The program next reads and validates the command input file. Errors are reported if invalid commands or command arguments are given or if required commands are omitted. If errors are detected, the program terminates with a non-zero status code.
- The program then does its thing. If errors are detected, the data base is "rolled back" to its previous, pre-program condition, and the program terminates with a non-zero status code. You may then correct the errors and run the program again.
- Upon successful completion, the elapsed time and a zero status code are displayed, the log file is closed, and the program terminates.

ICE Tool Kit Program Summary

The following table lists and briefly describes each ICE Tool Kit program. Individual programs are more completely documented in the remainder of this section.

Program	Program Description
ACTLIST	Lists all or a subset of the activities within a project.
ACTSTORE	Reads an activity attribute and MOSS EXPORT file pair, rasterizes the features and their zones-of-influence, and stores them and their schedules in a project's map cabinet.
BARCHART	Creates a bar chart comparing the HV, HE, or MRI values of 1 to 6 calendars for a project unit. Requires Quattro Pro.
CALDROP	Deletes a unit calendar from a project.
CALLIST	Lists all or a subset of unit calendars within a project.
CELLCNTS	Displays the current number of available and used slots in a project's cell map file.
CELLSLOT	Displays the feature slots at a specified cell location within a project.
COMPARE	Displays a comparison of HV, HE, and MRI values by period between two calendars for a project unit.
FEATDROP	Deletes a set of features from a project's map cabinet.
FEATLIST	Lists all or a subset of the features within a project.
HABSTORE	Reads a habitat reclass and MOSS EXPORT file pair, rasterizes the habitat features (vegetation, supplement, or unit boundary), and stores them in a project's map cabinet.
ICICLE	Creates an ICE_PROJECT table in the ORACLE data base.
MAPDROP	Deletes a set of maps and their features from a project's map cabinet. (Features on the map table are not deleted!)
MAPLIST	Lists all or a subset of the maps imported into a project via the ACTSTORE or HABSTORE programs.
PROJDROP	Deletes a project and all its supporting data entries and structures.
PROJLIST	Lists all or a subset of projects from the ICE_PROJECT table.

Program	Program Description
PROJNEW	Creates a new set of data base tables and files for a new project.
SCHEDULE	Reschedules the occurrence and disturbance-mortality classes of activity features within a project.
SHUFFLE	Moves a set of project features: (1) from the map cabinet onto the map table, (2) from the map table into the map cabinet, or (3) to different positions on the map table.
SNAPSHOT	Copies the "CURRENT" HV, HE, and MRI calendars (reflecting the current map composition) for a project unit under a new name for later use by BARCHART, COMPARE, CALLIST, and CALDROP.
UTM2CELL	Converts a UTM coordinate to a project's column and row number.

ACTLIST Program Reference

Description

ACTLIST is an ICE Tool Kit program to list a subset of activities within a project. The PROJECT command determines which project to access. An optional FEATURES command restricts the listing to a subset of activities meeting an SQL WHERE condition. Output may be in comma-delimited (CDF), form, table, or activity schedule format. Finally, output may optionally include activity schedule information.

Usage

```
ACTLIST  < command_input_file >  [ < log_file > ]
```

where < `command_input_file` > contains the program's input commands, and
[< `log_file` >] is an optional log file to capture console output.

Input Commands

Input Command Summary	
User <oracle_user_name>	Required
PAssword <oracle_password>	Required
PROJect <project_label>	Required
FEatures <sql_where_clause>	Optional
FOrmat { CDF | FORM | TABLE | SCHEDULE }	Optional
Verbose { NO | YES }	Optional

FEATURES < sql_where_clause>

FEATURES is an optional command used to restrict the activity listing to some subset of project activities. If omitted from the command input file, all activities are listed. The argument is a complete SQL WHERE clause beginning with "WHERE" and enclosed in quotes. The WHERE clause may reference any Feature Table columns. Some useful examples include:

```
1   FEATURE "WHERE FEATURE_NUMBER=1234"
2   FEATURE "WHERE FEATURE_NUMBER BETWEEN 1234 AND 2345"
3   FEATURE "WHERE FEATURE_GROUP LIKE '%A%'"
4   FEATURE "WHERE MAP_NUMBER = 34"
5   FEATURE "WHERE AREA > 1000"
```

WHERE clauses can become very elaborate and complex. Please refer to the ORACLE SQL*PLUS reference manual for more information on the SQL WHERE clause, and Chapter 6 of this documentation for the Feature Table column names.

FORMAT { CDF | FORM | TABLE | SCHEDULE }

FORMAT is an optional command that selects the format of the activity list. If omitted from the command input file, the FORM format is used. The format types are:

CDF	ASCII comma-delimited format, one activity per line.
FORM	Form style output, one activity per form.
TABLE	Table style output, one activity per table row.
SCHEDULE	ICE activity attribute file format.

VERBOSE { NO | YES }

The VERBOSE command indicates whether or not to include each activity's schedule in the output. If omitted from the command input file, YES is presumed.

Figure 2 illustrates a sample ACTLIST command input file.

```
* ICE ACTLIST Sample Input File
* Collin D. Bevins, 1991
*
* Display progress messages and sound bell when complet-
ed.
PROGRESS    YES
BELL        YES
*
* Identify my ORACLE user name and password.
USER        SYSTEM
PASSWORD    BANGKOK
*
* Identify the ICE project
PROJECT     SAMPLE
FEATURE    "WHERE FEATURE_NUMBER > 1"
VERBOSE     YES
```

Figure 2 - Sample ACTLIST command input file.

Output

Figure 3 illustrates output from the SAMPLE project ACTLIST command input file. This illustrates the FORM format with activity schedules listed ("VERBOSE YES").

Note that activities that are on the map table have a positive position, while those in the map cabinet have a negative map position.

```
--------------------------------------------------------------------------------
ICE ACTLIST Activity Listing Program   Version 0.80  Apr 91
Copyright (c) 1991 by Systems for Environmental Management
Started on Wed Jun 05 14:12:03 1991
--------------------------------------------------------------------------------


Activity : 8
Type     : Point
Label    : STORE
Descript : STORE/GAS STATION
Position : 1
Periods  : 1
0101-1231: P_M_H  P_A_F

Activity : 9
Type     : Linear
Label    : HIWAY 1:1
Descript : HIWAY 1 SEGMENT 1
Position : 2
Periods  : 1
0101-1231: L_M_H  L_H_F

Activity : 10
Type     : Linear
Label    : HIWAY 1:2
Descript : HIWAY 1 SEGMENT 2
Position : 2
Periods  : 1
0101-1231: L_M_H  L_H_F

Activity : 11
Type     : Linear
Label    : SPUR 1:1
Descript : SPUR 1 SEGMENT 1
Position : 3
Periods  : 1
0101-1231: L_M_L  L_L_F

Activity : 12
Type     : Dispersed
Label    : GRAZING
Descript : SHEEP ALLOTMENT
Position : 4
Periods  : 1
0301-0831: D_NM_L  D_L_F

Activity : 13
Type     : Dispersed
Label    : HUNTING
Descript : HUNTING/BIGHORNS
Position : 5
Periods  : 1
0901-1130: D_NM_H  D_H_F


6 activities were found.

--------------------------------------------------------------------------------
Completed with status 0 on Wed Jun 05 14:12:06 1991
Elapsed time = 2.3 seconds (0.0 min).
--------------------------------------------------------------------------------
```

Figure 3 - Sample ACTLIST output in FORM format.

ACTSTORE Program Reference

Description

ACTSTORE is an ICE Tool Kit program to store activity schedule and map information in a project's map cabinet. The program matches each feature from the MOSS EXPORT map file with the next ICE activity attribute record block, validates the inputs, rasterizes the activity, creates the zone-of-influence, and stores it. The SHUFFLE program must be used to move activities onto the map table.

Before running ACTSTORE, you must have a MOSS EXPORT file containing the point, linear, or polygon activity features (see Section 5.5). You must also have created an activity attribute file describing the occurrence and disturbance-mortality classes for each activity (see Section 5.1). Activities must occur in the same sequence in both files.

ACSTORE is a two-pass program. The first pass reads and validates the MOSS EXPORT and activity attribute files, creates the zones-of-influence, and rasterizes the activities. If errors are detected, a message is displayed and the program terminates. No changes will have been made to the data base. You may then correct the errors and run ACTSTORE again.

The second pass reads the rasterized map data and stores it in the data tables (e.g., in the map cabinet).

Usage

```
ACTSTORE  < command_input_file >  [ < log_file > ]
```

where < command_input_file > contains the program's input commands, and
[< log_file >] is an optional log file to capture console output.

Input Commands

Input Command Summary	
User <oracle_user_name>	Required
PAssword <oracle_password>	Required
PROJect <project_label>	Required
Map <moss_export_map_file>	Required
ATtributes <activity_attribute_file>	Required
Description <activity_map_description>	Required
Theme { POINT | LINEAR | POLYGON }	Required

5. DATA FILE FORMATS

ICE Tool Kit programs read and process eight types of data files;

Activity Attribute Files	Text files describing activity feature dates and displacement-mortality classes (read by the ACTSTORE program).
Activity Schedule Files	Text files describing activity feature replacement dates and disturbance-mortality classes (read by the SCHEDULE program.
Cell Map Files	Binary map files containing lists of features at each cell location.
Habitat Reclass Files	Text files describing reclassification rules from MOSS EXPORT subjects to ICE attribute classes (read by the HABSTORE program).
MOSS EXPORT Files	Text files containing feature subjects and coordinates (read by the ACTSTORE and HABSTORE programs).
ORACLE Data Base Files	ORACLE proprietary data base files containing ICE project tables (described in Section 6).
Program Input Files	Text files containing command-argument pairs (read by the ICE Tool Kit programs).
SQL*LOADER Control Files	Text files containing SQL*LOADER scripts to initialize the Attribute, Attribute Date, Displacement, Displacement-Cover, Edge Transition, Mortality, Mortality-Cover, Period, and Theme Tables.

5.1 Activity Attribute Files

Activity attribute files contain the labels, descriptions, dates, and disturbance and mortality classes for every activity in a MOSS EXPORT file. Activity attribute files are created with a text editor and are read by the ACTSTORE program.

There is one activity attribute file for every MOSS EXPORT activity map file. Within the activity attribute file, there is one activity attribute record block for every feature in the MOSS EXPORT file. The first activity attribute record block describes the first MOSS EXPORT feature, the second attribute block describes the second MOSS feature, and so forth.

The first record of the attribute file must contain the text "ICE ACTIVITY ATTRIBUTES". It is followed by one or more activity attribute record blocks.

An activity attribute record block

- begins with a single ACTIVITY record,
- is followed by zero or more DATES records, and
- terminates with a single ACCEPT-ACCEPTSAME-REJECT record.

Blank and commented lines may be interspersed throughout the file to improve readability and documentation. Comment lines must begin with an asterisk.

```
ICE ACTIVITY ATTRIBUTES
*
* SAMPLE Linear Activity Attributes
*
* First activity is placed at map table position 1
ACTIVITY  'HIWAY 1:1'  'A'  'HIWAY 1 SEGMENT 1'
DATES  0101  1231  L_M_H  L_H_F
ACCEPT
*
* Next activity is actually a continuance of the first,
* so it will also be placed at map table position 1
ACTIVITY  'HIWAY 1:2'  'A'  'HIWAY 1 SEGMENT 2'
DATES  0101  1231  L_M_H  L_H_F
ACCEPTSAME
*
* The last activity is placed at map table position 2.
ACTIVITY  'SPUR 1:1'  'A'  'SPUR 1 SEGMENT 1'
DATES  0101  1231  L_M_L  L_L_F
ACCEPT
```

Figure 1 - Sample activity attribute file.

Attribute records begin with a keyword and are followed by zero or more fields. All records are free-format; record fields must be separated by one or more blanks or tabs. Fields containing embedded blanks must be enclosed in single or double quotes.

ACTIVITY Record

The ACTIVITY record identifies each activity. There is exactly one ACTIVITY record for every feature in the MOSS EXPORT file. The ACTIVITY record contains four fields:

1 Field 1 is the keyword "ACTIVITY".

2 Field 2 is a unique 1 to 16-character activity label. The activity is indexed and identified by this label. A label could be descriptive, such as "HIWAY 287", or could indicate the activity's quadrangle-feature lineage, such as "BIGSP123". If the label includes a blank, comma, or any non-alphanumeric character, it must be inclosed in single or double quotes.

3 Field 3 is a 0 to 16-character activity group label. Activity groups are useful in identifying activity alternatives. A group label containing the letter "A", for example, could indicate activities in the current situation, while "B" indicates a proposed alternative. An activity with group label "AB" would then occur in both situations.

If no group label is desired, the field must contain two consecutive quotes (""). If the label includes a blank, comma, or any non-alphanumeric character, it must be enclosed in single or double quotes.

4 Field 4 is a 0 to 64-character activity description. If no description is desired, the field must contain two consecutive quotes (""). If the description includes a blank, comma, or any non-alphanumeric character, it must be enclosed in single or double quotes.

DATES Record

The DATES record identifies a period during which the activity has a certain disturbance and mortality class combination. An activity may have 0 to 365 DATES records. If no DATES records are given, the activity's description and location is stored in the data base; even though it never occurs, it may be rescheduled later.

The DATES record contains five fields:

1 Field 1 contains the keyword "DATES".

2 Field 2 contains the beginning date for the disturbance-mortality class combination. It is in "mmdd" format, where "0101" is January 1, "1231" is December 31, and "0524" is Bob Dylan's birthday.

3 Field 3 contains the ending date (inclusive) for the disturbance-mortality class.

4 Field 4 contains the disturbance class label for the period within the beginning and ending dates. The label must exist in the Displacement Table.

5 Field 5 contains the mortality class label for the period within the beginning and ending dates. The label must exist in the Mortality Table.

If DATES records have overlapping periods, latter definitions overlay earlier definitions. If an activity has the following DATES records, for example,

```
DATES  0101  0731  L_M_H  L_A_F
DATES  0601  1231  L_M_L  L_NA_F
```

it has an L_M_H disturbance during 0101-0531 (not 0101-0731), and an L_M_L disturbance during 0701-1231.

ACCEPT - ACCEPTSAME - REJECT Record

The final record in the activity attribute block tells ACTSTORE how to dispose of the activity. The record has a single field consisting of one of the three keywords "ACCEPT", "ACCEPTSAME", or "REJECT".

The REJECT keyword tells ACTSTORE to discard the activity. The activity feature, description, and dates are not stored in the ICE data base. REJECT is useful if a MOSS EXPORT file contains unwanted activity features, but you'd rather not delete them from the file.

The ACCEPT keyword tells ACTSTORE to rasterize and store the feature, its description, and dates in the ICE data base. As each activity is stored, it is assigned the next highest position on the map table.

Use of ACCEPT and ACCEPTSAME

The ICE SHUFFLE and SCHEDULE programs determine the impact of activities on the underlying habitat. The programs consider the impacts of activities *located at different positions on the map table to be cumulative.*

The ACCEPT keyword tells ACTSTORE to store an activity feature at the next highest position on the map table. Thus each activity normally occupies its own map table position, and its impacts are therefore cumulative.

In certain circumstances, an activity may actually be composed of two or more features. An example is a stretch of highway that has been arbitrarily broken into shorter segments, perhaps at quadrangle boundaries. If stored using the ACCEPT keyword, each segment is treated as an individual activity, resulting in unwanted cumulative impacts around their juncture.

In such a case, the segments should be sequentially ordered in the MOSS EXPORT and attribute files. The first feature of the activity should use the ACCEPT keyword, thereby storing it at a new map table position. The activity's subsequent features, however, should use the ACCEPTSAME keyword to store them at the same map table position as the first segment.

When SHUFFLE or SCHEDULE evaluate the activity's impact, they will find multiple features at that position. For each day of the year, the activity's feature with the greatest impact is selected. These daily values are then included in the cumulative impact calculations with the activities from the other map table positions.

The ACCEPTSAME keyword tells ACTSTORE to store the activity at the same map table position as the previous activity. This allows activities that have been artificially divided into smaller pieces, such as road segments, to be aggregated as a single, multi-featured activity.

Interaction with the ACTSTORE Program

The ACTSTORE program reads and validates both the MOSS EXPORT and activity attribute files before storing any activities in the map cabinet. It ensures that MOSS EXPORT files are intact and polygons are closed, that there is a one-to-one correspondence between MOSS EXPORT features and attribute record blocks, and validates disturbance and mortality class labels. Errors are reported to the screen and log file; you may then correct the data and re-run ACTSTORE.

5.2 Activity Schedule Files

Activity schedule files contain activity replacement schedules for input to the SCHEDULE program. SCHEDULE reads the activity schedule file, reschedules selected activities according to the file specifications, and recalculates the resulting activity HV and MRI impacts.

The activity schedule file is an ASCII text file you create with a text editor. It must contain the text "ICE ACTIVITY SCHEDULE" on the first record.

The file contains one or more activity schedule record blocks. An activity schedule record block

- begins with a single ACTIVITY record,
- is followed by zero or more DATES records, and
- terminates with a single UPDATE-NOUPDATE record.

```
ICE ACTIVITY SCHEDULE
*
* Open activity 12 for the entire year.
ACTIVITY "WHERE FEATURE_NUMBER=12"
DATES  0101  1231  L_M_L  L_L_F
UPDATE
*
* See what happens if FSR 1234 closed
ACTIVITY "WHERE FEATURE_LABEL LIKE 'FSR 1234%'"
NOUPDATE
```

Figure 2 - Sample activity schedule file.

Blank and commented lines may be interspersed throughout the file to improve readability and documentation. Comment lines must begin with an asterisk.

Activity schedule records begin with a keyword and are followed by zero or more fields. All records are free-format; record fields must be separated by one or more blanks or tabs. Fields containing embedded blanks must be enclosed in single or double quotes.

ACTIVITY Record

The ACTIVITY record specifies the set of activities to be rescheduled. The record contains two fields:

1 Field 1 is the keyword "ACTIVITY".

2 Field 2 is an SQL WHERE clause specifying the set of activities to be rescheduled. The WHERE clause:

- must begin with the SQL keyword "WHERE",
- must be a valid ORACLE SQL*PLUS WHERE clause,
- must reference Feature Table columns only, and
- must be enclosed in double quotes.

The SCHEDULE program parses the WHERE clause and submits it to ORACLE for further processing in a SELECT statement. All activities meeting the condition are selected and their schedules are replaced with the schedule given by the DATES records.

DATES Record

The DATES record identifies a period during which the selected activities will have a certain disturbance and mortality class combination. It has the same format as the activity attribute file DATES record (see section 5.1).

The existing schedule of each activity selected by the WHERE clause is erased and replaced by the schedule specified by the DATES records. If the schedule record block contains no DATES records, the selected activities are effectively removed from the landscape and never impact the habitat. This provides an alternative to using SHUFFLE to remove activities.

UPDATE-NOUPDATE Record

The final record in the activity schedule block tells SCHEDULE whether to make the schedule changes permanent. The record has a single field consisting of the keyword "UPDATE" or the keyword "NOUPDATE".

If "UPDATE" is specified, the new schedule is made permanent for all the activities selected by the WHERE clause.

If "NOUPDATE" is specified, the impact of the new schedule is calculated and displayed for each activity selected by the WHERE clause *one at a time*, and then the activity's original schedule is restored. If the WHERE clause selects two activities, for example, the new schedule's impact is calculated for the first activity alone, then for the second activity alone, *the impact is not calculated for both activities simultaneously.*

Interaction with SCHEDULE Program

SCHEDULE is the only program that reads and validates activity schedule files. It processes one schedule record block at a time. If an error is found in a record block, a message is displayed to the screen and the log file and the program terminates. You may then correct the problem, comment out (or delete) all the schedule blocks preceding the erroneous block, and re-run the program.

5.3 Cell Map Files

Cell map files are binary direct-access data files containing a list of features at each cell location. Each project has its own cell map file created by the PROJNEW program. Information is written into the file by the ACTSTORE, HABSTORE, and SHUFFLE programs. Information is read from the file by the SHUFFLE and SCHEDULE programs.

The ICE programs CELLCNT and CELLSLOT allow you to peek at the file. While ICE Tool Kit users never need to access this file, it is documented here for completeness.

The cell map file is a matrix with the same dimensions as its project rows and columns. It also has a depth controlled by the PROJNEW INITIAL command; if "INITIAL 32" is specified, then the map is 32 feature slots deep when created.

Each slot is a structure containing three signed four-byte integers:
- a feature number,
- the feature's attribute number, and
- the feature's vertical position on the map table.

The very first slot (slot 0) at each cell location is special in that it contains:
- the cell row,
- the cell column, and
- the number of features at the cell (not including slot 0).

ACTSTORE and HABSTORE read map features and rasterize them into cells. Each cell is stored at its appropriate cell map location in the next available slot. If no more slots are available at a cell location, the cell map file is expanded by INCREMENT <n> slots. If <n> is 8, the cell map will be 40 slots deep after expansion. If the slots at any cell location fill again, the file is again expanded, this time to 48 slots per cell.

Cell map file size can be calculated as:

```
file_bytes = project_rows x project_columns x feature_slots x 12
```

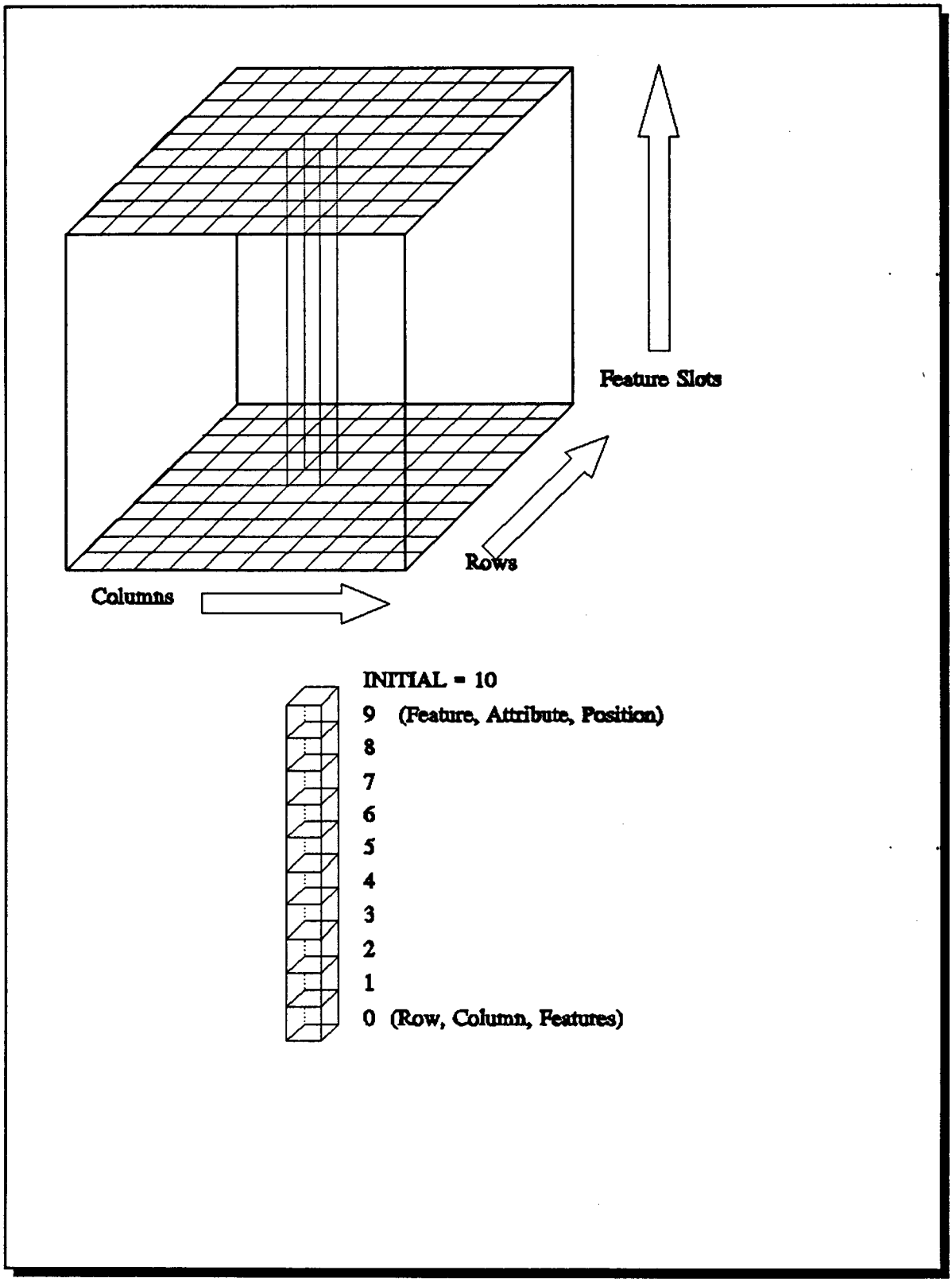

Figure 3 - ICE project cell map file structure.

A Note on Cell Map Attribute Numbers

The attribute numbers stored at each cell map slot are 6 digits long. The first two digits are the feature's theme id number from the Theme Table. The interpretation of the remaining four digits depends on the feature's theme.

For features with a habitat theme (vegetation, supplemental, or unit boundary), the last four digits are its attribute class from the Attribute Table.

For point, linear, and dispersed activities, the last four digits indicate the distance in meters from the cell location to the nearest source cell for the activity.

A Note of Cell Map Position Numbers

The position numbers stored at each cell map slot indicate the feature's current or potential position on the map table.

If the position is less than zero, the feature is currently in the map cabinet, and not on the map table. When the feature is SHUFFLEd onto the map table, the position is updated to its absolute value.

If the position is greater than zero, the feature is currently on the map table, and not in the map cabinet. When the feature is SHUFFLEd back into the map cabinet, the position is updated to its negative value.

5.4 Habitat Reclass Files

Habitat reclass files are used by the HABSTORE program to translate MOSS EXPORT subjects into CEM attributes. They are ASCII text files you create with a text editor. A separate habitat reclass file is required for

- vegetation habitat features,
- supplemental habitat features, and
- unit boundary features.

Normally a single set of 3 habitat reclass files is adequate for an entire ecosystem.

The first record of a habitat reclass file must contain the text "ICE RECLASS". It is followed by up to 1800 reclass records. Each reclass record contains two fields:

1 Field 1 is a MOSS EXPORT subject exactly as it appears in a map file. If the subject contains embedded blanks, it must be enclosed in quotes.

2 Field 2 is a valid vegetation, supplement, or unit boundary class label from the Attribute Table. If this field is "NONE", features with the matching subject are not stored in the map cabinet.

The file may also contain blank or comment lines to improve readability and documentation. Comment lines must begin with an asterisk.

As HABSTORE reads each vegetation, supplement, or unit boundary feature from the MOSS EXPORT file, it searches the reclass file for the feature's subject. If not found, an error message is displayed to the console and the log file. You must then add the subject to the reclass file and re-run HABSTORE.

MOSS EXPORT - Habitat Reclass - Attribute Table File Relationships

```
  1         72044                                  5
000000.00 0000500.00
000000.00 0001000.00
001000.00 0001000.00
001000.00 0000500.00
000000.00 0000500.00
  2         00011                                  5
000000.00 0000000.00
000000.00 0000500.00
001000.00 0000500.00
001000.00 0000000.00
000000.00 0000000.00
```

Figure 4 - MOSS EXPORT File

```
ICE RECLASS
*
*MOSS EXPORT Subject---------  Attribute Class
00011                          V:X:L
0001172044                     V:X:LH
7204400011                     V:X:HL
72044                          V:X:H
CLEARCUT                       V:L:R
*
* Ignore features with NU subject
NU                             NONE
*
* Enclosed subjects with blanks inside quotes.
"POND OR LAKE"                 V:N:L
```

Figure 5 - Habitat Reclass File

```
.
.
.
310401  V:X:L
310402  V:X:LH
310403  V:X:HL
310404  V:X:H
310405  V:X:R
.
.
.
```

Figure 5 - Project Attribute Table

5.5 MOSS EXPORT Files

MOSS EXPORT files are ASCII text files describing the content and location of map features. The ICE ACTSTORE and HABSTORE programs read features from MOSS EXPORT files; all your map data must be fed into ICE using this file format. MOSS EXPORT files may be created with a text editor, generated from the MOSS GIS using its EXPORT program, or from the ARC/INFO GIS using its ARCMOSS program.

A MOSS EXPORT file contains one or more features. Each feature consists of a header record and one or more coordinate pair records.

MOSS EXPORT Header Records

The header record contains three fields:

- The *item* field is the feature's item (sequence) number as assigned by the GIS.
- The *subject* is a 0 to 30-character feature label.
- The *number of points* field is the number of coordinate pair records that follow.

Table 1 - MOSS EXPORT header record format.

MOSS EXPORT Header Record Format				
Field	Type	Length	Begin Column	End Column
Item number	Integer	5	1	5
Subject	Character	30	16	45
Number of coordinate pair records	Integer	5	51	55

MOSS EXPORT Coordinate Pair Records

The coordinate pair records contain two fields, the UTM easting or x-coordinate, and the UTM northing or y-coordinate.

MOSS EXPORT files contain either point, linear, or polygon features. Point activities are point features and have only one coordinate pair per feature. Linear activities are linear features with two or more coordinate pairs per feature.

Dispersed activities, vegetation, supplement, and unit boundary features are all polygon features with four or more coordinate pairs. The first and last coordinate pair must be identical to ensure polygon closure.

Polygon features may contain any number of embedded islands. The islands are enclaves with undefined subjects (which may be defined and filled by subsequent features). If a feature has islands, the first block of coordinate pairs specify the outer perimeter, which must be closed and progress in a clockwise direction. The outer perimeter coordinate record blocks are followed by any number of island coordinate record blocks.

The first coordinate pair of an island is flagged with a negative x-coordinate. The remaining coordinate pairs identify the island perimeter, which must be closed. The last coordinate pair for the island must be identical to the islands first coordinate pair (to ensure closure), except that it has no negative sign.

The points field in the header record must include all the outer perimeter and island coordinate pairs.

Table 2 - MOSS EXPORT coordinate pair record format.

MOSS EXPORT Coordinate Pair Record Format				
Field	Type	Length	Begin Column	End Column
X coordinate (UTM easting) (A negative value flags the start of a polygon island.)	Fixed decimal	11.2	1	11
Y coordinate (UTM northing)	Fixed decimal	11.2	12	22

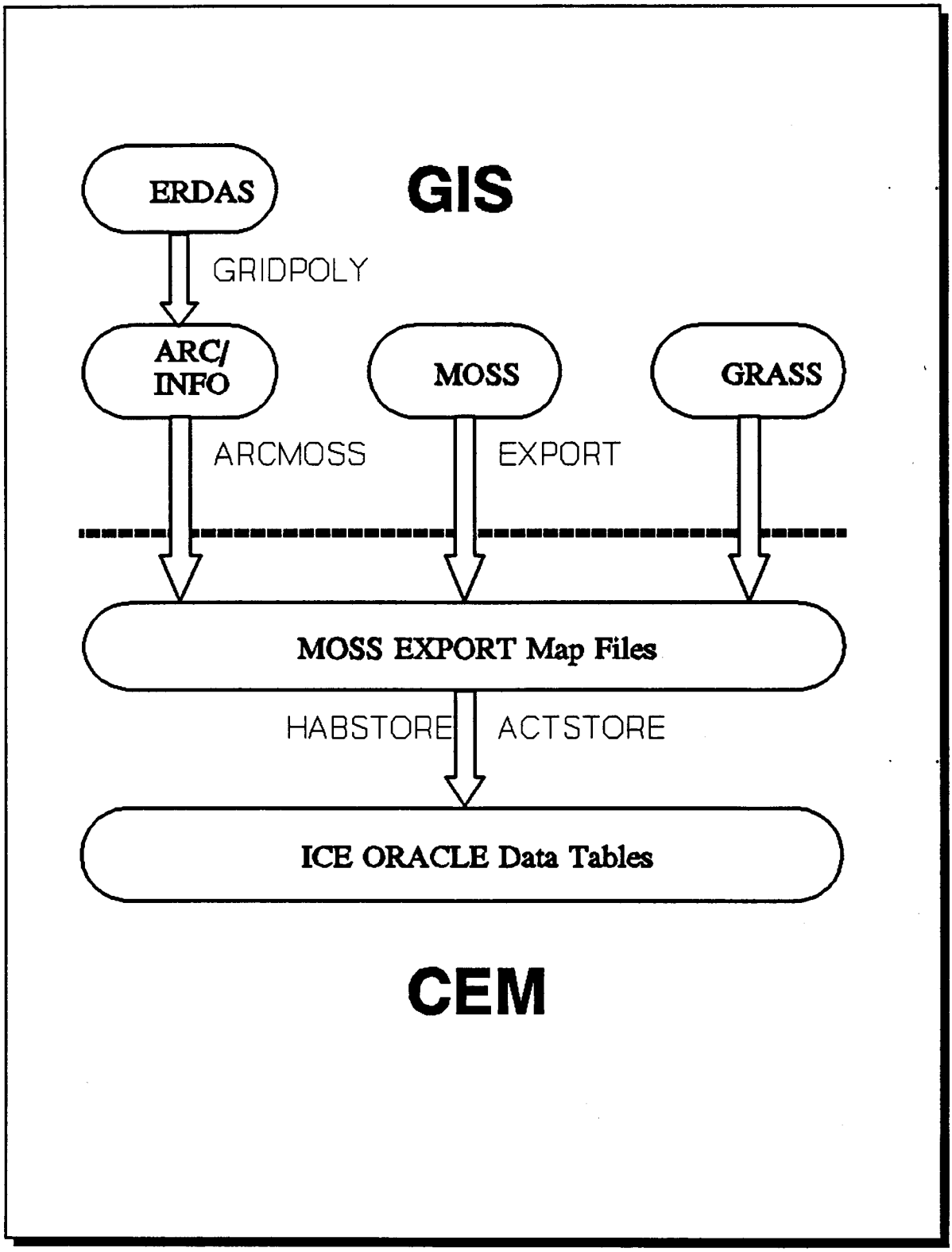

Figure 7 - The ICE Tool Kit data files are separate from the GIS; MOSS EXPORT files provide the common data interface.

5.6 ORACLE Data Base Files

The ORACLE relational data base consists of two files on your computer; the data base file (usually named "dbs.ora") and the before image file ("bi.ora"). The location and names of these two files are specified to ORACLE in the initialization file "init.ora", which is usually located in "\oracle5\dbs\init.ora".

Both files are black boxes within which the relational data base infrastructure and the ICE tables, clusters, and indices are defined. If you are brave of heart, you can access the data base to examine table definitions and contents using the SQL*PLUS interactive query language. Chapter 6 documents all the tables created and maintained by ICE.

ORACLE MACHINE_TYPE

If you are running ORACLE on the MS-DOS operating system, and have the QEMM memory manager installed, you must install ORACLE using MACHINE_TYPE J.

ORACLE System Buffers

To maximize the performance of ORACLE on your MS-DOS machine, include the following statement in the "oracle5\dbs\init.ora" initialization file:

```
buffers 890
```

Creating a Separate CEM Data Base

While the ORACLE installation procedure creates a default data base, you may wish to create a separate data base just for CEM. This procedure is detailed in Appendix C of *ORACLE Database for MS-DOS User's Guide*, and is paraphrased here for your convenience.

1 Make sure ORACLE is running on the computer:
```
C> ORACLE
```

2 Use ORACLE's EXP utility to export the entire default data base to a dump file:
```
C> EXP SYS sys_password
```

3 Create a directory where you want the new data base files and move into that directory:
```
C> E:
E> MD \GRIZ
E> CD \GRIZ
```

4 Run ORACLE's CCF utility to create a contiguous data base file of the size you need. To create a 100 megabyte file named DBS.ORA:
```
E> CCF DBS.ORA 204800
```

5 Run ORACLE's CCF utility to create a contiguous before image file of the size you need (about 20% the size of the data base file):
```
E> CCF BI.ORA 41000
```

6 Edit the "database" and "before_image" entries in the ORACLE initialization file (usually at "\oracle5\dbs\init.ora") to read:
```
DATABASE        E:\GRIZ\DBS.ORA
BEFORE_IMAGE    E:\GRIZ\BI.ORA
```

7 Warm start ORACLE with the initialization flag I:
```
C> IOR I
```

8 Run the ORACLE IMP utility to import the dump file you created in step 2:
```
C> IMP SYS password
```

9 Change passwords and grant privileges:
```
C> SQL*PLUS SYS/sys_password
> GRANT CONNECT TO SYS IDENTIFIED BY <new_sys_password>;
> GRANT CONNECT SYSTEM/system_password;
> GRANT CONNECT TO SYSTEM IDENTIFIED BY <new_system_password>;
> EXIT
```

5.7 Program Input Files

Running ICE Programs

All ICE Tool Kit programs read their run time commands from an input file. The programs are invoked by entering the program name, the command input file name, and an optional log file name on the system command line. For example,

```
D> PROJNEW SAMPLE.INP SAMPLE.LOG
```

invokes the PROJNEW program, which will read its input commands from the file "SAMPLE.INP", and record its output in the log file "SAMPLE.LOG". If the log file doesn't exist, it is created. If the log file already exists, it will be appended.

You can always get a brief description of an ICE program and its input commands by entering just the program name on the system command line. For example,

```
D> PROJNEW
```

causes the display in figure 8 to be printed to your console.

ICE Input Command File Format

Program input files are ASCII text files you create and modify with a text editor. A file may contain blank lines, comment lines, and command lines (figure 9).

Comment lines begin with an asterisk and may contain any text.

Command lines contain two fields. The first field is a valid ICE program keyword. The second field is the command argument. If the second field contains embedded blanks or non-alphanumeric characters, it must be enclosed in quotes. The keyword and arguments are free-field; they must be separated by one or more blanks or tabs.

```
--------------------------------------------------------------------------------
ICE PROJNEW Project Creation Program   Version 0.80  Apr 91
Copyright (c) 1991 by Systems for Environmental Management
--------------------------------------------------------------------------------

USAGE:
------
PROJNEW   <input_file>  [<log_file>]
where     <input_file> contains the program input commands, and
          <log_file>   is an optional log file to capture output.

DESCRIPTION:
------------
PROJNEW allows you to create a new ICE project.
You define the project area in the <input_file>, and PROJNEW stores
the definitions and creates the necessary data base tables.
NOTE: After creating the new project, you should run SQLLOAD to load
project parameters into the newly created tables!!

INPUTS:
-------
User              Req  Data base user name
PAssword          Req  Data base user password
PROJect           Req  Project label (1-8 chars)
Description       Opt  Project description (0-64 chars)
ECOsystem         Opt  Ecosystem label (0-16 chars)
SPecies           Opt  Species label (0-16 chars)
CEllmap           Req  Cell map file (full pathname) (1-80 chars)
INItial           Opt  Initial slots per cell (default=32)
INCrement         Opt  Cell map slot increment size (default=8)
Zone              Req  UTM Zone number
WEst              Req  Project's western-most boundary UTM
SOuth             Req  Project's southern-most boundary UTM
EAst              Req  Project's eastern-most boundary UTM
North             Req  Project's northern-most boundary UTM
WIDth             Req  Cell width (meters)
Height            Req  Cell height (meters)
Mri*hv            Opt  Multiply HV into MRI value (YES or NO)
SUpplement        Opt  Supplement habitat valuation (CUMULATIVE, MAXIMUM, or TOP)

--------------------------------------------------------------------------------
```

Figure 8 - All ICE programs display a screen of information when just the program name is entered on the command line. The display above is from the PROJNEW program.

```
* ICE PROJNEW Sample Input File to create the SAMPLE
project.
* Collin D. Bevins, 1991
*
* Display progress messages and sound bell when complet-
ed.
PROGRESS    YES
BELL        YES
*
* Identify my ORACLE user name and password.
USER        SYSTEM
PASSWORD    BANGKOK
*
* Sample project definitions.
PROJECT     SAMPLE
DESCRIPTION "Sample Project Area"
ECOSYSTEM   "Sample Ecosystem"
SPECIES     "Grizzly Bear"
*
* Define the cell map file and its initial and incremen-
tal feature slots.
CELLMAP     d:\gye\tiny\maps\sample.map
INITIAL     12
INCREMENT   4
*
* Define project window.
ZONE        12
WEST        0.
SOUTH       0
EAST        999
NORTH       999.
*
* Define the cell resolution.
WIDTH       100
HEIGHT      100.
*
* Specify whether to include HV in the MRI calculations.
MRI*HV      NO
*
* Treat supplemental habitat coefficients as cumulative.
SUPPLEMENT  CUMULATIVE
```

Figure 9 - Sample PROJNEW command input file.

Required ICE Command Keywords

The following ICE command keywords are required by all ICE programs:

USER	Identifies your ORACLE user name.
PASSWORD	Identifies your ORACLE password.
PROJECT	Identifies the project the program is to work on.

Optional ICE Command Keywords

The following optional ICE command keywords are available to all ICE programs.

BELL	If YES, the console's bell rings when the program terminates.
ECHO	If YES, the input file commands are echoed to the console and log file as they are processed.
PROGRESS	If YES, intermediate results are displayed to the console and the log file. While this allows you to track the program's progress, it can lead to large log files.

In addition, each ICE program has its own set of valid input commands, both required and optional. These are fully documented in Chapter 4.

5.8 SQL*LOADER Control Files

The PROJNEW program creates empty ORACLE tables for a new ICE project. You must use the ORACLE SQL*LOADER utility to load your ecosystem parameters into the new tables.

The SQL*LOADER program (SQLLOAD) reads its control information and your ecosystem parameters from a *control file*. The control file has two sections. The first section tells SQLLOAD, among other things, in which tables to store the data. The second section contains the table data.

```
-- SQL*LOADER control file to load an empty Theme Table.
-- You must modify the table name in line 12 to match the project label.
-- Collin D. Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE THEME.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_THEME
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(THEME_NUMBER, THEME_LABEL, THEME_DESCRIPTION, THEME_ZONE, THEME_TOP_POSITION)
--
BEGINDATA
11, "COVER",      "Security Cover",                    0, 0
12, "EDGE",       "Habitat Edge",                    100, 0
21, "UNIT",       "Unit Boundary",                     0, 0
31, "VEGETATION", "Vegetation: Habitat & Cover Type",  0, 0
32, "SUPPLEMENT", "Supplemental Protein Source",       0, 0
41, "POINT",      "Point Activity",                 1000, 0
42, "LINEAR",     "Linear Activity",                1000, 0
43, "DISPERSED",  "Dispersed Activity",                0, 0
```

Figure 10 - Sample THEME.CTL control file to load a project Theme Table.

An ecosystem usually has 9 SQL*LOADER control files, one for each project data table to be loaded. Before running SQLLOAD for a new project, you must edit the project name, usually on line 12 of each control file, with the new project name.

The project name is the prefix to the project's table names. In the figure above, if the new project name is "MADU2", the table name "SAMPLE_THEME" must be changed to "MADU2_THEME".

The figure illustrates several characteristics of a control file:

- Comment lines must begin with a double dash ("--").
- The control section beings with the "LOAD DATA" command and includes the table name and its column names.
- The "REPLACE" command tells SQLLOAD to delete the table's current contents before inserting the new data.
- The data section begins with the command "BEGINDATA".
- Comments are not allowed after the BEGINDATA command.
- There is one data record for each table row.
- Data fields follow the order specified in the control section.
- All data fields are separated by commas.
- Character fields are enclosed in quotes.

SQL*LOADER commands, keywords, and formats are documented in *SQL*LOADER User's Guide*, part of your ORACLE documentation set.

Running SQLLOAD

To run SQLLOAD, you must specify your ORACLE name, password, and the control file on the command line:

```
C:> SQLLOAD SYSTEM/PASSWORD ATTR.CTL
```

SQLLOAD reads the control file and creates a discard file, a reject file, and a log file. For the above example, these files are ATTR.DSC, ATTR.BAD, and ATTR.LOG. If SQLLOAD was successful, the ATTR.DSC and ATTR.BAD files will have 0 bytes. The ATTR.LOG file gives the gory details of how all the records were loaded and any errors detected.

A batch file, LOADPARM.BAT, is provided to run SQLLOAD for each of the nine control files in the proper sequence. You will need to edit this file with your own ORACLE user name and password:

SQLLOAD SYSTEM/PASSWORD ATTR.CTL
SQLLOAD SYSTEM/PASSWORD ATTRDATE.CTL
SQLLOAD SYSTEM/PASSWORD DISP.CTL
SQLLOAD SYSTEM/PASSWORD DISPCOVR.CTL
SQLLOAD SYSTEM/PASSWORD EDGETRAN.CTL
SQLLOAD SYSTEM/PASSWORD MORT.CTL
SQLLOAD SYSTEM/PASSWORD MORTCOVR.CTL
SQLLOAD SYSTEM/PASSWORD PERIOD.CTL
SQLLOAD SYSTEM/PASSWORD THEME.CTL

The remainder of this section provides guidelines for creating the nine standard ICE control files.

ATTR.CTL - Attribute Table Control File

The project's Attribute Table contains the ecosystem's class definitions for unit boundaries, vegetation, and supplemental habitats. It also defines the security cover and edge classes.

The Attribute Table is loaded from the ATTR.CTL control file. There is one record for every class. Each record has five fields:

attribute_number	A unique 6-digit class id number. The first two digits must be a valid theme number from the Theme Table. The remaining four digits may be assigned as needed.
attribute_label	A unique 1 to 16-character class label. By convention, the first two characters should be "C:" for cover, "E:" for edge, "S:" for supplement, "U:" for unit boundary, or "V:" for vegetation.
attribute_description	A 0 to 64-character description of the class.
attribute_display	Not used.
attribute_xref	A 6-digit class id number. This is used only by vegetation classes to indicate its security cover class. This number must be a valid security cover class id number.

The sample ATTR.CTL on the next page contains recommended security cover and edge classes, and demonstrates how vegetation classes are cross-referenced to security cover classes.

Note that the Attribute Table must be loaded from ATTR.CTL **before** the Attribute Date Table is loaded from ATTRDATE.CTL.

```
-- SQL*LOADER control file to load habitat classes into Attribute Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE ATTR.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_ATTRIBUTE
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(ATTRIBUTE_NUMBER, ATTRIBUTE_LABEL, ATTRIBUTE_DESCRIPTION,
ATTRIBUTE_DISPLAY, ATTRIBUTE_XREF)
--
BEGINDATA
110001, "C:L",       "Low Security Cover",                                  "..", 0
110002, "C:LH",      "Mosaic with Predominantly Low Cover",                 "::", 0
110003, "C:HL",      "Mosaic with Predominantly High Cover",                "00", 0
110004, "C:H",       "High Security Cover",                                 "88", 0
110005, "C:R",       "Removed Cover",                                       "xx", 0
120000, "E:N",       "No Edge",                                             "..", 0
120001, "E:L",       "Low cover side of edge",                              "nn", 0
120002, "E:H",       "High cover side of edge",                             "NN", 0
210001, "U:UNIT1",   "Unit 1 - Covers Entire Area",                         "UU", 0
210002, "U:SUBUNIT1","Sub Unit 1 of Unit 1",                                "S1", 0
310101, "V:L:L",     "Low Habitat Value with Low Cover",                    "..",  110001
310102, "V:L:LH",    "Low Habitat Value with Low & High Cover",             "..",  110002
310103, "V:L:HL",    "Low Habitat Value with High & Low Cover",             "..",  110003
310104, "V:L:H",     "Low Habitat Value with High Cover",                   "..",  110004
310105, "V:L:R",     "Low Habitat Value with Removed Cover",                "..",  110005
310201, "V:M:L",     "Moderate Habitat Value with Low Cover",               "..",  110001
310202, "V:M:LH",    "Moderate Habitat Value with Low & High Cover",        "..",  110002
310203, "V:M:HL",    "Moderate Habitat Value with High & Low Cover",        "..",  110003
310204, "V:M:H",     "Moderate Habitat Value with High Cover",              "..",  110004
310205, "V:M:R",     "Moderate Habitat Value with Removed Cover",           "..",  110005
310301, "V:H:L",     "High Habitat Value with Low Cover",                   "..",  110001
310302, "V:H:LH",    "High Habitat Value with Low & High Cover",            "..",  110002
310303, "V:H:HL",    "High Habitat Value with High & Low Cover",            "..",  110003
310304, "V:H:H",     "High Habitat Value with High Cover",                  "..",  110004
310305, "V:H:R",     "High Habitat Value with Removed Cover",               "..",  110005
310401, "V:X:L",     "Extremely High Habitat Value with Low Cover",         "..",  110001
310402, "V:X:LH",    "Extremely High Habitat Value with Low & High Cover",  "..",  110002
310403, "V:X:HL",    "Extremely High Habitat Value with High & Low Cover",  "..",  110003
310404, "V:X:H",     "Extremely High Habitat Value with High Cover",        "..",  110004
310405, "V:X:R",     "Extremely High Habitat Value with Removed Cover",     "..",  110005
320001, "S:BISON_WINTER", "Bison Winter Range",                             "BB", 0
320002, "S:CARION",       "Carion Concentrations",                          "CC", 0
320003, "S:ELK_HIGH",     "Higher Elevation Mountainous Elk Winter Range",  "HH", 0
320004, "S:ELK_LOW",      "Lower Elevation Valley Elk Winter Range",        "LL", 0
320005, "S:ELK_THERMAL",  "Geothermally Influenced Elk Winter Range",       "TT", 0
320006, "S:SPAWN_HIGH",   "High Value Fish Spawning Stream",                "FF", 0
320007, "S:SPAWN_LOW",    "Low Value Fish Spawning Stream",                 "ff", 0
320008, "S:MOOSE_WINTER", "Moose Winter Range",                             "MM", 0
320009, "S:UNG_CALVING",  "Ungulate Calving Grounds",                       "uu", 0
320010, "S:UNG_SUMMER",   "Ungulate Summer-Fall High Use Range",            "UU", 0
```

Figure 11 - Sample ATTR.CTL control file to load a project Attribute Table.

ATTRDATE.CTL - Attribute Date Table Control File

The Attribute Date Table contains the ecosystem's seasonal values for the edge, vegetation, and supplemental habitat classes defined in the Attribute Table. The Attribute Date Table is loaded from the ATTRDATE.CTL control file.

Each edge, vegetation, or supplemental class defined in the Attribute Table may have zero, one, or multiple entries in the Attribute Date Table. There is one record for each season of each class. Each record has four fields:

attribute_number	A 6-digit attribute id number from the Attribute Table.
attribute_begins	A 4-digit date in "mmdd" format indicating the first day of a period or season.
attribute_ends	A 4-digit date in "mmdd" format indicating the last day of the period or season.
attribute_value	A number (with or without a decimal point) indicating the class value or coefficient during the period or season.

Note that the Attribute Date Table must be loaded from ATTRDATE.CTL **after** the Attribute Table has been loaded from ATTR.CTL.

```
-- SQL*LOADER control file to load habitat classes into Attribute Date Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE ATTRDATE.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_ATTRIBUTE_DATE
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(ATTRIBUTE_NUMBER, ATTRIBUTE_BEGINS, ATTRIBUTE_ENDS, ATTRIBUTE_VALUE)
--
BEGINDATA
120000,  0101,  1231,   1.0
120001,  0101,  1231,   2.0
120002,  0101,  1231,   3.0
310101,  0101,  1231,  0.0001
310102,  0101,  1231,  0.0001
310103,  0101,  1231,  0.0001
310104,  0101,  1231,  0.0001
310105,  0101,  1231,  0.0001
310201,  0101,  1231,  0.0100
310202,  0101,  1231,  0.0100
310203,  0101,  1231,  0.0100
310204,  0101,  1231,  0.0100
310205,  0101,  1231,  0.0100
310301,  0101,  1231,  0.1000
310302,  0101,  1231,  0.1000
310303,  0101,  1231,  0.1000
310304,  0101,  1231,  0.1000
310305,  0101,  1231,  0.1000
310401,  0101,  1231,  0.5000
310402,  0101,  1231,  0.5000
310403,  0101,  1231,  0.5000
310404,  0101,  1231,  0.5000
310405,  0101,  1231,  0.5000
320001,  0101,  1231,  1.
320002,  0101,  1231,  1.
320003,  0101,  0228,  1.
320003,  0301,  0531,  1.6
320003,  0601,  1231,  1.
320004,  0101,  1231,  1.
320005,  0101,  0228,  1.
320005,  0301,  0531,  1.9
320005,  0601,  1231,  1.
320006,  0101,  0531,  1.
320006,  0601,  0831,  3.75
320006,  0901,  1231,  1.
320007,  0101,  0531,  1.
320007,  0601,  0831,  1.9
320007,  0901,  1231,  1.
320008,  0101,  1231,  1.
320009,  0101,  1231,  1.
320010,  0101,  0531,  1.
320010,  0601,  1130,  1.5
320010,  1201,  1231,  1.
```

Figure 12 - Sample ATTRDATE.CTL control file to load a project Attribute Date Table.

DISP.CTL - Displacement Table Control File

The Displacement Table defines the ecosystem's disturbance classes and their zone-of-influence radius and decay method. The Displacement Table is loaded from the DISP.CTL control file. There is one record for each disturbance class. The record has five fields:

displacement_number	A unique 2-digit disturbance class id number.
displacement_label	A unique 1 to 16-character disturbance class label.
displacement_description	A 0 to 64-character description of the disturbance class.
displacement_zone	The zone-of-influence (meters) for the disturbance class. (1 mile = 1610 m, 0.5 mile = 805 m, 0.25 mile = 403 m).
displacement_decay	An integer indicating how disturbance impacts decay with increasing distance from the activity. Use "0" for constant impact (no decay over distance), "1" for inversely proportional to distance from the activity, or "2" for inversely proportional to the distance from the activity squared.

Note that the Displacement Table must be loaded from DISP.CTL **before** the Displacement Cover Table is loaded from DISPCOVR.CTL.

```
-- SQL*LOADER control file to load the Displacement Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE DISP.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_DISPLACEMENT
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(DISPLACEMENT_NUMBER, DISPLACEMENT_LABEL, DISPLACEMENT_DESCRIPTION,
DISPLACEMENT_ZONE, DISPLACEMENT_DECAY)
--
BEGINDATA
01, "AIRCRAFT",   "Aircraft",                          1000,  2
02, "D_M_H",      "Dispersed Motorized High Use",         0,  2
03, "D_M_L",      "Dispersed Motorized Low Use",          0,  2
04, "D_NM_H",     "Dispersed Non-motorized High Use",     0,  2
05, "D_NM_L",     "Dispersed Non-motorized Low Use",      0,  2
06, "EXPLOSIVES", "Explosives",                        1000,  2
07, "L_M_H",      "Linear Motorized High Use",         1000,  2
08, "L_M_L",      "Linear Motorized Low Use",          1000,  2
09, "L_M_I",      "Linear Motorized Incidental Use",   1000,  2
10, "L_NM_H",     "Linear Non-motorized High Use",      500,  2
11, "L_NM_L",     "Linear Non-motorized Low Use",       500,  2
12, "P_M_24",     "Point Motorized 24 Use",            1000,  2
13, "P_M_H",      "Point Motorized High Diurnal Use",  1000,  2
14, "P_M_L",      "Point Motorized Low Diurnal Use",   1000,  2
15, "P_NM_24",    "Point Non-motorized 24 Use",         500,  2
16, "P_NM_D",     "Point Non-motorized Diurnal Use",    500,  2
```

Figure 13 - Sample DISP.CTL control file to load a project Displacement Table.

DISPCOVR.CTL - Displacement Cover Table Control File

The Displacement Cover Table contains the disturbance coefficients under various security cover conditions for each Displacement Table class. The Displacement Cover Table is loaded from the DISPCOVR.CTL control file.

The Displacement Cover Table contains one entry for each disturbance class - security class combination. Since there are 16 disturbance classes and 5 security cover classes in the illustrated samples, there must be 80 entries in the Displacement Cover Table. Each record has four fields:

displacement_number	A 2-digit disturbance class code from the Displacement Table.
cover_number	A 6-digit security cover class code from the Attribute Table.
parm1	The disturbance coefficient at the activity source. This must be in the range [0,1] where 1 signifies no impact, and 0 is total impact.
parm2	The disturbance coefficient at the edge of the zone-of-influence. This must be in the range [0,1] where 1 signifies no impact, and 0 is total impact.

Note that the Displacement Cover Table must be loaded from DISPCOVR.CTL **after** the Displacement Table has been loaded from DISP.CTL.

```
-- SQL*LOADER control file to load the Displacement Cover Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE DISPCOVR.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_DISPLACEMENT_COVER
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(DISPLACEMENT_NUMBER, COVER_NUMBER, PARM1, PARM2)
--
BEGINDATA
01,  110001,  0.3,  1.0
01,  110002,  0.4,  1.0
01,  110003,  0.5,  1.0
01,  110004,  0.6,  1.0
01,  110005,  0.3,  1.0
02,  110001,  0.3,  1.0
02,  110002,  0.4,  1.0
02,  110003,  0.5,  1.0
02,  110004,  0.6,  1.0
02,  110005,  0.3,  1.0
03,  110001,  0.4,  1.0
03,  110002,  0.5,  1.0
03,  110003,  0.6,  1.0
03,  110004,  0.7,  1.0
03,  110005,  0.4,  1.0
04,  110001,  0.5,  1.0
04,  110002,  0.6,  1.0
04,  110003,  0.7,  1.0
04,  110004,  0.8,  1.0
04,  110005,  0.5,  1.0
05,  110001,  0.6,  1.0
05,  110002,  0.7,  1.0
05,  110003,  0.8,  1.0
05,  110004,  0.9,  1.0
05,  110005,  0.6,  1.0
06,  110001,  0.3,  1.0
06,  110002,  0.4,  1.0
06,  110003,  0.5,  1.0
06,  110004,  0.6,  1.0
06,  110005,  0.3,  1.0
07,  110001,  0.2,  1.0
07,  110002,  0.3,  1.0
07,  110003,  0.4,  1.0
07,  110004,  0.5,  1.0
07,  110005,  0.2,  1.0
08,  110001,  0.3,  1.0
08,  110002,  0.4,  1.0
08,  110003,  0.5,  1.0
08,  110004,  0.6,  1.0
08,  110005,  0.3,  1.0
09,  110001,  0.4,  1.0
09,  110002,  0.5,  1.0
09,  110003,  0.6,  1.0
09,  110004,  0.7,  1.0
09,  110005,  0.4,  1.0
10,  110001,  0.5,  1.0
```

Figure 14 - Sample DISPCOVR.CTL control file to load a project Displacement Cover Table.

EDGETRAN.CTL - Edge Transition Table Control File

The Edge Transition Table contains the rules for determining a cell's edge type. The Edge Transition Table is loaded from the EDGETRAN.CTL control file.

The ICE SHUFFLE and SCHEDULE programs automatically determine the type of edge for each cell from the rules in the Edge Transition Table. The table below illustrates the edge transition logic for the sample project.

The security cover class for the cell under examination is located in the first column. Let's assume the cell has a high cover type, C:H. The program then checks all cells within the edge buffer distance of this cell. If the first neighbor has a security type of C:H, the result is no edge (E:N). If another neighbor has security cover type C:HL, then the source cell is in the high side of an edge buffer (E:H). By definition, mosaic cells (C:LH and C:HL) are always in an edge buffer. Similarly, cells with removed cover (C:R) are considered "man-caused" and don't contribute to edge.

The logic of the edge transition matrix is entered into the EDGETRAN.CTL control file and loaded into the Edge Transition Table. The table has three fields per record:

Table 3 - Edge transition matrix for the Sample project.

Source Cell Security Cover	Neighboring Cell Security Cover				
	C:L 110001	C:LH 110002	C:HL 110003	C:H 110004	C:R 110005
C:L 110001	E:N 120000	E:L 120001	E:L 120001	E:L 120001	E:N 120000
C:LH 110002	E:L 120001	E:L 120001	E:L 120001	E:L 120001	E:L 120001
C:HL 110003	E:H 120002	E:H 120002	E:H 120002	E:H 120002	E:H 120002
C:H 110004	E:H 120002	E:H 120002	E:H 120002	E:N 120000	E:N 120000
C:R 110005	E:N 120000	E:N 120000	E:N 120000	E:N 120000	E:N 120000

source_cover_number	The source cell's 6-digit security class id number from the Attribute Table.
neighbor_cover_number	The neighboring cell's 6-digit security class id number from the Attribute Table.
edge_number	The resulting edge class 6-digit id number from the Attribute Table.

Note that the Edge Transition Table must be loaded from EDGETRAN.CTL **after** the Attribute Table has been loaded from ATTR.CTL.

```
-- SQL*LOADER control file to load the Edge Transition Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE EDGETRAN.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_EDGE_TRANSITION
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(SOURCE_COVER_NUMBER, NEIGHBOR_COVER_NUMBER, EDGE_NUMBER)
--
BEGINDATA
110001, 110001, 120000
110001, 110002, 120001
110001, 110003, 120001
110001, 110004, 120001
110001, 110005, 120000
110002, 110001, 120001
110002, 110002, 120001
110002, 110003, 120001
110002, 110004, 120001
110002, 110005, 120001
110003, 110001, 120002
110003, 110002, 120002
110003, 110003, 120002
110003, 110004, 120002
110003, 110005, 120002
110004, 110001, 120002
110004, 110002, 120002
110004, 110003, 120002
110004, 110004, 120000
110004, 110005, 120000
110005, 110001, 120000
110005, 110002, 120000
110005, 110003, 120000
110005, 110004, 120000
110005, 110005, 120000
```

Figure 15 - Sample EDGETRAN.CTL control file to load a project Edge Transition Table.

MORT.CTL - Mortality Table Control File

The Mortality Table defines the ecosystem's mortality classes and their zone-of-influence radius and decay method. The Mortality Table is loaded from the MORT.CTL control file. There is one record for each mortality class. The record has five fields:

mortality_number	A unique 2-digit mortality class id number.
mortality_label	A unique 1 to 16-character disturbance class label.
mortality_description	A 0 to 64-character description of the mortality class.
mortality_zone	A zone-of-influence (meters) for the mortality class. (1 mile = 1610 m, 0.5 mile = 805 m, 0.25 mile = 403 m).
mortality_decay	An integer indicating how mortality effects decay with increasing distance from the activity. Use "0" for constant effect (no decay over distance), "1" for inversely proportional to distance from the activity, or "2" for inversely proportional to the distance from the activity squared.

Note that the Mortality Table must be loaded from MORT.CTL **before** the Mortality Cover Table is loaded from MORTCOVR.CTL.

```
-- SQL*LOADER control file to load the Mortality Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE MORT.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_MORTALITY
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(MORTALITY_NUMBER, MORTALITY_LABEL, MORTALITY_DESCRIPTION,
MORTALITY_ZONE, MORTALITY_DECAY)
--
BEGINDATA
01,  "D_H_F",    "Dispersed High Use, Firearms",         0,  2
02,  "D_H_NF",   "Dispersed High Use, No Firearms",      0,  2
03,  "D_L_F",    "Dispersed Low Use, Firearms",          0,  2
04,  "D_L_NF",   "Dispersed Low Use, No Firearms",       0,  2
05,  "L_H_F",    "Linear High Use, Firearms",         1000,  2
06,  "L_H_NF",   "Linear High Use, No Firearms",      1000,  2
07,  "L_L_F",    "Linear Low Use, Firearms",          1000,  2
08,  "L_L_NF",   "Linear Low Use, No Firearms",       1000,  2
09,  "P_A_F",    "Point Attractant, Firearms",        1000,  2
10,  "P_A_NF",   "Point Attractant, No Firearms",     1000,  2
11,  "P_NA_F",   "Point No Attractants, Firearms",    1000,  2
12,  "P_NA_NF",  "Point No Attractants, No Firearms", 1000,  2
```

Figure 16 - Sample MORT.CTL control file to load a project Mortality Table.

MORTCOVR.CTL - Mortality Cover Table Control File

The Mortality Cover Table contains the mortality coefficients under various security cover conditions for each Mortality Table class. The Mortality Cover Table is loaded from the MORTCOVR.CTL control file.

The Mortality Cover Table contains one entry for each mortality class - security class combination. Since there are 12 mortality classes and 5 security cover classes in the illustrated samples, there must be 60 entries in the Mortality Cover Table. Each record has four fields:

mortality_number	A 2-digit mortality class code from the Mortality Table.
cover_number	A 6-digit security cover class code from the Attribute Table.
parm1	The mortality coefficient at the activity source, where 0 indicates no mortality.
parm2	The mortality coefficient at the edge of the zone-of-influence, where 0 indicates no mortality.

Note that the Mortality Cover Table must be loaded from MORTCOVR.CTL **after** the Mortality Table has been loaded from MORT.CTL.

```
-- SQL*LOADER control file to load the Mortality Cover Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE MORTCOVR.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_MORTALITY_COVER
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(MORTALITY_NUMBER, COVER_NUMBER, PARM1, PARM2)
--
BEGINDATA
01,  110001,  0.9,  0.0
01,  110002,  0.8,  0.0
01,  110003,  0.7,  0.0
01,  110004,  0.6,  0.0
01,  110005,  0.9,  0.0
02,  110001,  0.5,  0.0
02,  110002,  0.4,  0.0
02,  110003,  0.3,  0.0
02,  110004,  0.2,  0.0
02,  110005,  0.5,  0.0
03,  110001,  0.7,  0.0
03,  110002,  0.6,  0.0
03,  110003,  0.5,  0.0
03,  110004,  0.4,  0.0
03,  110005,  0.7,  0.0
04,  110001,  0.4,  0.0
04,  110002,  0.3,  0.0
04,  110003,  0.2,  0.0
04,  110004,  0.1,  0.0
04,  110005,  0.4,  0.0
05,  110001,  0.9,  0.0
05,  110002,  0.8,  0.0
05,  110003,  0.7,  0.0
05,  110004,  0.6,  0.0
05,  110005,  0.9,  0.0
06,  110001,  0.5,  0.0
06,  110002,  0.4,  0.0
06,  110003,  0.3,  0.0
06,  110004,  0.2,  0.0
06,  110005,  0.5,  0.0
07,  110001,  0.7,  0.0
07,  110002,  0.6,  0.0
07,  110003,  0.5,  0.0
07,  110004,  0.4,  0.0
07,  110005,  0.7,  0.0
08,  110001,  0.4,  0.0
08,  110002,  0.3,  0.0
08,  110003,  0.2,  0.0
08,  110004,  0.1,  0.0
08,  110005,  0.4,  0.0
09,  110001,  0.9,  0.0
09,  110002,  0.8,  0.0
09,  110003,  0.7,  0.0
09,  110004,  0.6,  0.0
09,  110005,  0.9,  0.0
10,  110001,  0.5,  0.0
```

Figure 17 - Sample MORTCOVR.CTL control file to load a project Mortality Cover Table.

PERIOD.CTL - Period Table Control File

The Period Table contains the beginning and ending dates for each of the project's periods or seasons. The Period Table is loaded from the PERIOD.CTL control file. There is one record per period, and each record has five fields:

period_number	A unique 3-digit period id number.
period_begins	The first date for the period in "mmdd" format.
period_ends	The last day of the period in "mmdd" format.
period_label	A 1 to 16-character period label.
period_description	A 0 to 64-character description of the period.

```
-- SQL*LOADER control file to load the Period Table.
-- You must modify the table name in line 12 to match the project label.
-- C.Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE PERIOD.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_PERIOD
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(PERIOD_NUMBER, PERIOD_BEGINS, PERIOD_ENDS, PERIOD_LABEL, PERIOD_DESCRIPTION)
--
BEGINDATA
1,  0301,  0531,  "Spring",  "Spring Grizzly Habitat Season"
2,  0601,  0831,  "Summer",  "Summer Grizzly Habitat Season"
3,  0901,  1130,  "Fall",    "Fall Grizzly Habitat Season"
```

Figure 18 - Sample PERIOD.CTL control file to load a project Period Table.

THEME.CTL - Theme Table Control File

The Theme Table defines the CEM themes. It is loaded from the THEME.CTL control file. The table has exactly 8 entries. Each entry has five fields:

theme_number — A unique 2-digit theme id number. **The ICE theme numbers are fixed and hard-wired into the model code -- DO NOT CHANGE THEM!!**

theme_label — A unique 1 to 16-character theme label.

theme_description — A 0 to 64-character theme description.

theme_zone — For the point and linear themes, this is the maximum zone-of-influence (meters) for any disturbance or mortality class. All activities will be rasterized to this zone width.

For the edge theme, this is the edge buffer distance. If you want cells on the diagonal to be within a source cell's edge zone, you must adjust this value accordingly! This field is unused by the other themes.

theme_top_position — Used internally by the ICE programs; **do not change this field!**

```
-- SQL*LOADER control file to load an empty Theme Table.
-- You must modify the table name in line 12 to match the project label.
-- Collin D. Bevins, 1991
--
LOAD DATA
INFILE *
DISCARDFILE THEME.DSC
REPLACE
--
-- Replace "SAMPLE" with the project label.
--
INTO TABLE SAMPLE_THEME
FIELDS TERMINATED BY "," OPTIONALLY ENCLOSED BY '"'
(THEME_NUMBER, THEME_LABEL, THEME_DESCRIPTION, THEME_ZONE, THEME_TOP_POSITION)
--
BEGINDATA
11, "COVER",      "Security Cover",                    0, 0
12, "EDGE",       "Habitat Edge",                    100, 0
21, "UNIT",       "Unit Boundary",                     0, 0
31, "VEGETATION", "Vegetation: Habitat & Cover Type",  0, 0
32, "SUPPLEMENT", "Supplemental Protein Source",       0, 0
41, "POINT",      "Point Activity",                 1000, 0
42, "LINEAR",     "Linear Activity",                1000, 0
43, "DISPERSED",  "Dispersed Activity",                0, 0
```

Figure 19 - Sample THEME.CTL control file to load a project Theme Table.

6. DATA BASE TABLES

The ICE Tool Kit serves two purposes. First, it provides a repository of CEM parameters, map data, and activity schedules. Second, it calculates HV, HE, and MRI values for various map table compositions.

Project data are stored in two places. The list of features occurring at each cell within the project is stored in the cell map file you specify during PROJNEW. The cell map file is a binary direct-access file maintained by the computer's operating system (see section 5.3). There is one cell map file for every project you create.

All other data are stored in the ORACLE relational data base manager. Once an ORACLE data base has been created on your computer, it will contain all project parameter, feature, and schedule information. Note that this single data base will contain data for all your projects.

A single ICE_PROJECT table exists within the data base. It contains the project definitions you supply to PROJNEW such as project label, description, cell resolution, and bounding UTMs.

In addition, each project has 14 data base tables to keep track of maps, features, activities, parameters, and calendars (see table 1). They have a standard base table name preceded by the project's unique 1-8 character label.

You can query and display the ICE data base tables using ORACLE's SQL*PLUS facility. You may also update tables using SQL*PLUS, although I strongly recommend you limit your updating to descriptive fields only.

Tables 2 through 16 document the ICE data base table structures and serve as your reference for performing SQL*PLUS queries.

Table 1: ICE Data Base Tables	
Table Name	**Table Contents**
Project Table ICE_PROJECT	Unique project label, description, location, and resolution for every project in the data base. One entry per project.
Activity Table project_ACTIVITY	Activity feature number, begin and end dates, and their disturbance and mortality codes. One or more entries per activity.
Attribute Table project_ATTRIBUTE	Unique id number, label, description, and cross-reference. One entry per cover, edge, unit, supplement, and vegetation class.
Attribute Date Table project_ATTRIBUTE_DATE	Attribute id, begin date, end date, and coefficient. One or more entries per Attribute Table entry.
Calendar table project_CALENDAR	Unique id number, unit id number, calendar label, and description. One entry per calendar.
Calendar Date Table project_CALENDAR_DATE	Calendar id number, period id number, and HV, HE, and MRI value. One entry per period per calendar.
Displacement Table project_DISPLACEMENT	Unique id number, label, description, zone width, and decay function. One entry per disturbance class.
Displacement Cover Table project_DISPLACEMENT_COVER	Displacement id number, cover class id number, and coefficients at activity source and periphery. One entry per security cover class per displacement class.
Edge Transition table project_EDGE_TRANSITION	Source cell cover id number, neighboring cell cover id number, and the resulting edge id number. One entry per cover-to-cover combination.
Feature Table project_FEATURE	Unique feature id number, attribute id number, feature label, description, source map number, area length, cells, and mbr. One entry per feature.
Map Table project_MAP	Unique map id number, theme, map file, attribute file, and imported features and cells. One entry per stored map.
Mortality Table project_MORTALITY	Unique id number, label, description, zone width, and decay functions. One entry per mortality class.
Mortality Cover Table project_MORTALITY_COVER	Mortality id number, cover class id number, and coefficients at activity source and periphery. One entry per security cover class per mortality class.
Period Table project_PERIOD	Unique period number, period begin and end dates, and period label and description. One entry per calendar period.
Theme Table project_THEME	Unique theme id, label, description, and zone width. One entry per theme.

Table 2: Activity Table		
Field	Type	Description
activity_number	N(6,0) INT4	Activity id number; same as its Feature.feature_number
activity_begins	N(4,0) INT2	First day (mmdd) the activity has this displacement and mortality type.
activity_ends	N(4,0) INT2	Last day (mmdd) the activity has this displacement and mortality type.
displacement_number	N(4,0) INT2	Id number of displacement class in effect between the dates.
mortality_number	N(4,0) INT2	Id number of mortality class in effect between the dates.

Indices

Indexed on activity_number (not unique)

Table Relationships

activity_number <<---> Feature.feature_number

displacement_number <<---> Displacement.displacement_number
mortality_number <<---> Mortality.mortality_number

Table 3: Attribute Table		
Field	**Type**	**Description**
attribute_number	N(6,0) INT4	Unique attribute id number in "ttaaaa" format where "tt" is a theme id number and "aaaa" is a unique theme class number.
attribute_label	C(16)	Unique attribute class label for cover, edge, extra, supplement, unit, or vegetation class.
attribute_description	C(64)	Attribute description.
attribute_display	C(2)	ASCII characters used to display features with this attribute.
attribute_xref	N(6,0) INT4	Cross reference to another attribute entry. Used to associate each vegetation attribute with a security cover attribute.

Indices

Clustered with Attribute Date Table on attribute_number

Table Relationships

attribute_number <---> > Attribute_Date.attribute_number
attribute_number <---> > Feature.attribute_number
attribute_xref < <---> Attribute.attribute_number

attribute_number <---> > Calendar.unit_number (unit attributes only)
attribute_number <---> > Edge.source_cover_number (cover attributes only)
attribute_number <---> > Edge.neighbor_cover_number (cover attributes only)
attribute_number <---> > Edge.edge_number (edge attributes only)
attribute_number <---> > Displacement.displacement_cover (cover attributes only)
attribute_number <---> > Mortality.mortality_cover (cover attributes only)

Table 4: Attribute Date Table		
Field	**Type**	**Description**
attribute_number	N(6,0) INT4	Attribute id number referencing the Attribute Table.
attribute_begins	N(4,0) INT2	First day (mmdd) the attribute has this value.
attribute_ends	N(4,0) INT2	Last day (mmdd) the attribute has this value.
attribute_value	N(8,5) FLOAT8	Attribute value within the two dates (inclusive).

Indices

Clustered with Attribute Table on attribute_number

Table Relationships

attribute_number <---> > Attribute.attribute_number
attribute_number <---> > Feature.attribute_number

attribute_number <---> > Calendar.unit_number (unit attributes only)
attribute_number <---> > Displacement.displacement_cover (cover attributes only)
attribute_number <---> > Mortality.mortality_cover (cover attributes only)

attribute_number <---> > Edge.source_cover_number (cover attributes only)
attribute_number <---> > Edge.neighbor_cover_number (cover attributes only)
attribute_number <---> > Edge.edge_number (edge attributes only)

Table 5: Calendar Table		
Field	Type	Description
calendar_number	N(4,0) INT2	Unique calendar id number.
unit_number	N(6,0) INT4	Unit id number referencing a unit in the Attribute Table.
calendar_label	C(16)	Unique calendar label.
calendar_description	C(64)	Calendar description.

Indices

Clustered with Calendar Date Table on calendar_number

Table Relationships

calendar_number <--->>> CalendarDate.calendar_number
unit_number <<---> Attribute.attribute_number

Note

Each unit has at least one entry in this table with a "CURRENT" calendar_label.
The SNAPSHOT program creates copies of the CURRENT calendar.

Table 6: Calendar Date Table		
Field	Type	Description
calendar_number	N(4,0) INT2	Calendar id number referencing the Calendar Table.
period_number	N(3,0) INT2	Period id number referencing the Period Table.
hv	N(12,2) FLOAT8	Sum of the period's habitat values for all cells within the unit.
he	N(12,2) FLOAT8	Sum of the period's habitat effectiveness for all cells within the unit.
mri	N(12,2) FLOAT8	Sum of the period's mortality risk index for all cells within the unit.

Indices

Clustered with Calendar Table on calendar_number

Table Relationships

calendar_number <<---> Calendar.calendar_number
period_number <<---> Period.period_number

Table 7: Displacement Table		
Field	Type	Description
displacement_number	N(4,0) INT2	Unique displacement id number.
displacement_label	C(16)	Unique displacement label.
displacement_ description	C(64)	Description of displacement class.
displacement_zone	N(4,0) INT2	Zone-of-influence radius (m).
displacement_decay	N(1,0) INT2	Method of determining displacement value at cells of known distance from the activity source: 0 = constant propagation 1 = inversely proportional to distance 2 = inversely proportional to distance squared

Indices

Clustered with Displacement Cover Table on displacement_number

Table Relationships

displacement_number <---> > DisplacementCover.displacement_number
displacement_number <---> > Activity.displacement_number

Note

This table extends potential displacement modelling options by:

1 allowing the displacement zone-of-influence to vary by date,
2 allowing displacement to vary by any number of cover situations, and
3 allowing displacement impacts to decline over distance from the activity source according to some predefined decay function.

Table 8: Displacement Cover Table		
Field	Type	Description
displacement_number	N(4,0) INT2	Displacement id number referencing the Displacement Table.
cover_number	N(6,0) INT4	Security cover id number referencing the Attribute Table.
parm1	N(8,6) FLOAT8	Displacement value at activity source.
parm2	N(8,6) FLOAT8	Displacement value at edge of activity zone-of-influence.

Indices

Clustered with Displacement Table on displacement_number

Table Relationships

displacement_number <<---> Displacement.displacement_number
displacement_number <<--->> Activity.displacement_number
cover_number <<---> Attribute.attribute_number (for cover attributes only)
cover_number <<--->> AttributeDate.attribute_number (for cover attributes only)

Note

This table extends potential displacement modelling options by:

1 allowing the displacement zone-of-influence to vary by date,
2 allowing displacement to vary by any number of cover situations, and
3 allowing displacement impacts to decline over distance from the activity source according to some predefined decay function.

Table 9: Edge Transition Table		
Field	Type	Description
source_cover_number	N(6,0) INT4	Security cover id number (referencing the Attribute Table) indicating the security cover class of a source cell.
neighbor_cover_ number	N(6,0) INT4	Security cover id number (referencing the Attribute Table) indicating the security cover class of a neighboring cell.
edge_number	N(6,0) INT4	Edge id number (referencing the Attribute Table) indicating the edge class associated with the source-to-neighbor transition.

Indices

Indexed on source_cover_number & neighbor_cover_number.

Table Relationships

source_cover_number <<---> Attribute.attribute_number (cover attributes only)
neighbor_cover_number <<---> Attribute.attribute_number (cover attributes only)
edge_number <<---> Attribute.attribute_number (edge attributes only)

Notes

This table defines the rules for determining the type of edge for a source cell. All neighboring cells within the edge buffer distance (defined in the Project Table) are examined for their cover type. The neighbor whose edge number has the highest edge value is selected for each day.

Table 10: Feature Table		
Field	**Type**	**Description**
feature_number	N(6,0) INT4	Unique feature id number.
attribute_number	N(6,0) INT4	Feature's 6-digit attribute id number from the Attribute Table. The first two digits are the feature's theme from the Theme Table. For habitat features, the last four digits are the feature's class id number from the Attribute Table. For activity features, the last four digits are "0000".
feature_label	C(16)	Feature label.
feature_group	C(16)	Feature group label.
feature_description	C(64)	Feature description.
feature_display	C(2)	ASCII characters used to display the feature on output maps.
map_number	N(4,0) INT2	Feature's map id number from the Map Table.
map_item	N(4,0) INT2	Feature's sequence number in the MOSS EXPORT map file.
current_position	N(6,0) INT4	Feature's vertical position on the map table. If less than zero, the feature is in the Map Cabinet. If greater than zero, the feature is on the Map Table.
new_position	N(6,0) INT4	Not used.
area	N(12,0) FLOAT8	Polygon size or point/linear activity zone size (whether inside or outside the project area).
length	N(8,0) FLOAT8	Polygon feature outside perimeter or linear feature length (whether inside or outside project area).
cells	N(10,0) INT4	Number of project cells covered by the feature or activity source cells within the project area.
west_cell	4x N(4,0) INT2	Feature's minimum bounding rectangle or mbr of activity zone of influence.
south_cell		
east_cell		
north_cell		

Table Relationships

feature_number <---> > Activity.activity_number (activity features only)
attribute_number < <---> Attribute.attribute_number

Table 11: Map Table		
Field	**Type**	**Description**
map_number	N(4,0) INT2	Unique map id number.
format_number	N(2,0) INT2	0=MOSS, 1=ARC, 2=ERDAS, 3=GRASS
theme_number	N(2,0) INT2	Theme id number referencing the Theme Table.
map_file	C(64)	Name of input file containing feature map data.
map_attribute_file	C(64)	Name of file containing feature attributes.
map_description	C(64)	Description of map area, contents, vintage, etc.
import_date_time	C(20)	Date and time of import run.
first_feature_number	N(6,0) INT4	First imported feature's id number.
last_feature_number	N(6,0) INT4	Last imported feature's id number.
imported_cells	N(10,0) INT4	Number of cells imported, or number of activity zone cells imported.
west_cell	4x N(4,0) INT2	Minimum bounding rectangle of all imported features or activity zones of influence.
south_cell		
east_cell		
north_cell		

Table Relationships

map_number <---> > Feature.map_number
theme_number < <---> Theme.theme_number
first_feature_number <---> Feature.feature_number
last_feature_number <---> Feature.feature_number

Table 12: Mortality Table		
Field	Type	Description
mortality_number	N(4,0) INT2	Unique mortality id number.
mortality_label	C(16)	Unique mortality label.
mortality_description	C(64)	Description of mortality class.
mortality_zone	N(4,0) INT2	Mortality zone-of-influence radius (m).
mortality_decay	N(1,0) INT2	Method of determining mortality value at cells of known distance from the activity source: 0 = constant propagation 1 = inversely proportional to distance 2 = inversely proportional to distance squared

Indices

Clustered with Mortality Cover Table on mortality_number

Table Relationships

mortality_number <--->> MortalityCover.mortality_number
mortality_number <--->> Activity.mortality_number

Note

This table extends potential mortality modelling options by:

1 allowing the mortality zone-of-influence to vary by date,
2 allowing mortality to vary by any number of cover situations, and
3 allowing mortality impacts to decline over distance from the activity source according to some predefined decay function.

Table 13: Mortality Cover Table		
Field	Type	Description
mortality_number	N(4,0) INT2	Mortality id number referencing the Mortality Table.
cover_number	N(6,0) INT4	Security cover id number referencing the Attribute Table.
parm1	N(8,6) FLOAT8	Mortality value at activity source.
parm2	N(8,6) FLOAT8	Mortality value at edge of activity zone-of-influence.

Indices

Clustered with Mortality Table on mortality_number

Table Relationships

mortality_number <<---> Mortality.mortality_number
mortality_number <<--->> Activity.mortality_number
cover_number <<---> Attribute.attribute_number (for cover attributes only)
cover_number <<--->> AttributeDate.attribute_number (for cover attributes only)

Note

This table extends potential mortality modelling options by:

1 allowing the mortality zone-of-influence to vary by date,
2 allowing mortality to vary by any number of cover situations, and
3 allowing mortality impacts to decline over distance from the activity source according to some predefined decay function.

Table 14: Period Table		
Field	Type	Description
period_number	N(3,0) INT2	Unique period id number.
period_begins	N(4,0) INT2	Date (mmdd) period begins.
period_ends	N(4,0) INT2	Date (mmdd) period ends.
period_label	C(16)	Unique period label.
period_description	C(64)	Period description.

Indices

Indexed on period_number

Table Relationships

period_number <---> > CalendarDate.period_number

Table 15: Project Table		
Field	Type	Description
project_label	C(8)	Unique label used as a prefix to the project's tables.
project_description	C(64)	Summary of analysis purpose and extent.
project_ecosystem	C(16)	Ecosystem name.
project_species	C(16)	Species name.
cell_file	C(80)	Complete pathname of project's cellmap file.
cell_initial	N(3,0) INT2	Initial cell slot allocation for the cellmap file.
cell_increment	N(3,0) INT2	Cellmap slot increment size.
utm_zone	N(2,0) INT2	Project's UTM zone.
utm_west	N(10,2) FLOAT8	Project's minimum bounding rectangle in UTMs.
utm_south		
utm_east		
utm_north		
cell_width	N(6,2) FLOAT8	Cell east-west resolution (m).
cell_height	N(6,2) FLOAT8	Cell north-south resolution (m).
mri_method	N(1,0) INT2	Mortality risk index computation method: 0 = mri not multiplied by hv 1 = mri multiplied by hv
supplement_method	N(1,0) INT2	0 = CUMULATIVE_MULTIPLICATIVE 1 = ADDITIVE 2 = MAXIMUM_BENEFIT 3 = HIGHEST_SUPPLEMENT
top_feature_number	N(6,0) INT4	Current highest feature sequence number.
top_map_number	N(4,0) INT2	Current highest import sequence number.

Table 16: Theme Table		
Field	Type	Description
theme_number	N(2,0) INT2	Unique theme id number.
theme_label	C(16)	Unique theme label.
theme_description	C(64)	Theme description.
theme_zone	N(4,0) INT2	Buffer zone width (m) for features of this theme. Used by edge and point, linear, and dispersed activity themes.
theme_top_position	N(6,0) INT4	Current highest feature position for this theme.

Indices

None.

Table Relationships

The theme_number is the first two digits of the six-digit attribute_number stored in the Attribute, AttributeDate, and Feature Tables.